Session 140

Darwin Kernel

Jim Magee
Member, Kernel Team
and
part-time MIG pilot

What You'll Learn

- The basic kernel services
- System layering above those services
- When it is "safe" to directly use kernel services
- Some alternatives to direct use
- How this reflects in debugging
- How this reflects in tuning

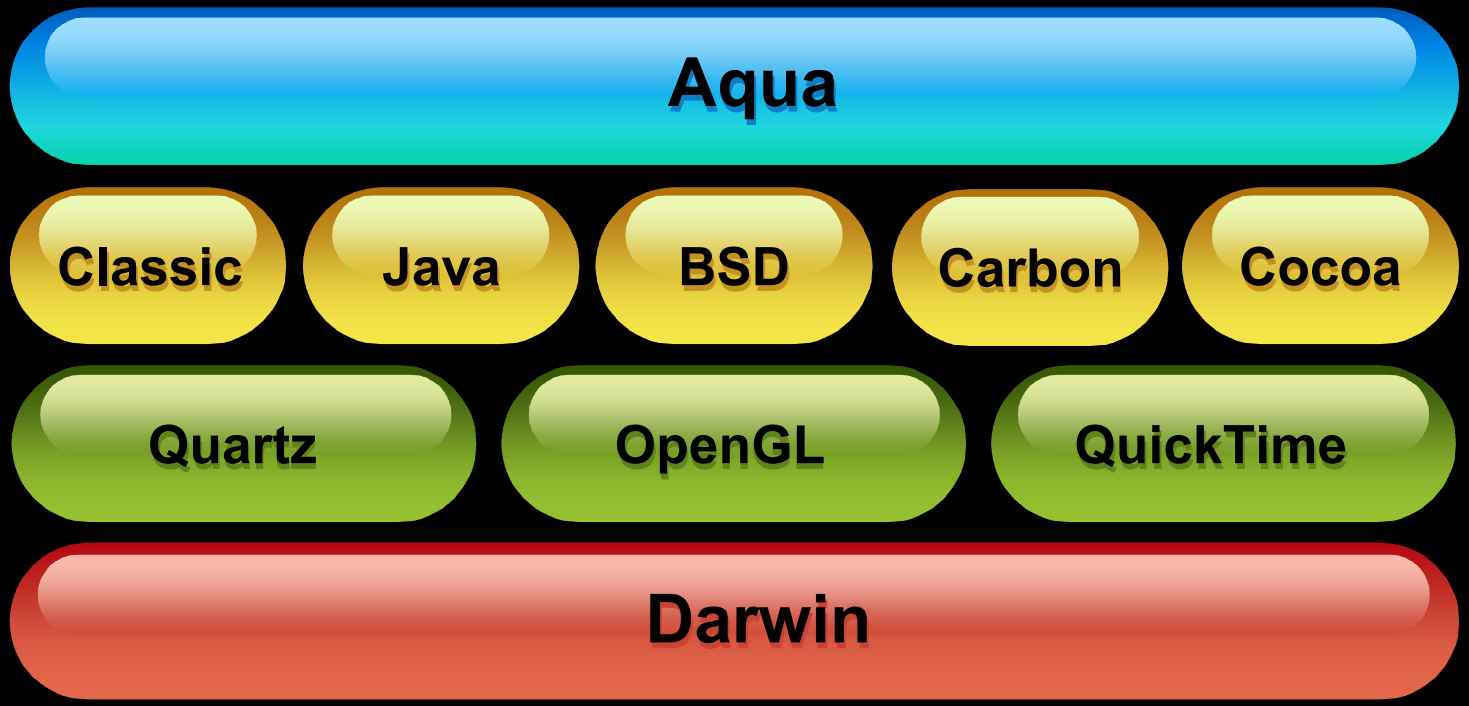
What You Won't Learn Here

- Filesystem implementation details
  - Or how to write a filesystem KEXT
- Networking implementation details
  - Or how to write a network KEXT
- Much about in-kernel programming
  - Although many of the same guidelines apply

Audience for This Session

- Applications developers
  - To get the most out of their programs
- KEXT developers
  - Who need a feel for the services available (without getting into too many details)
- Power users
  - To understand the demands on the system

Mac OS X

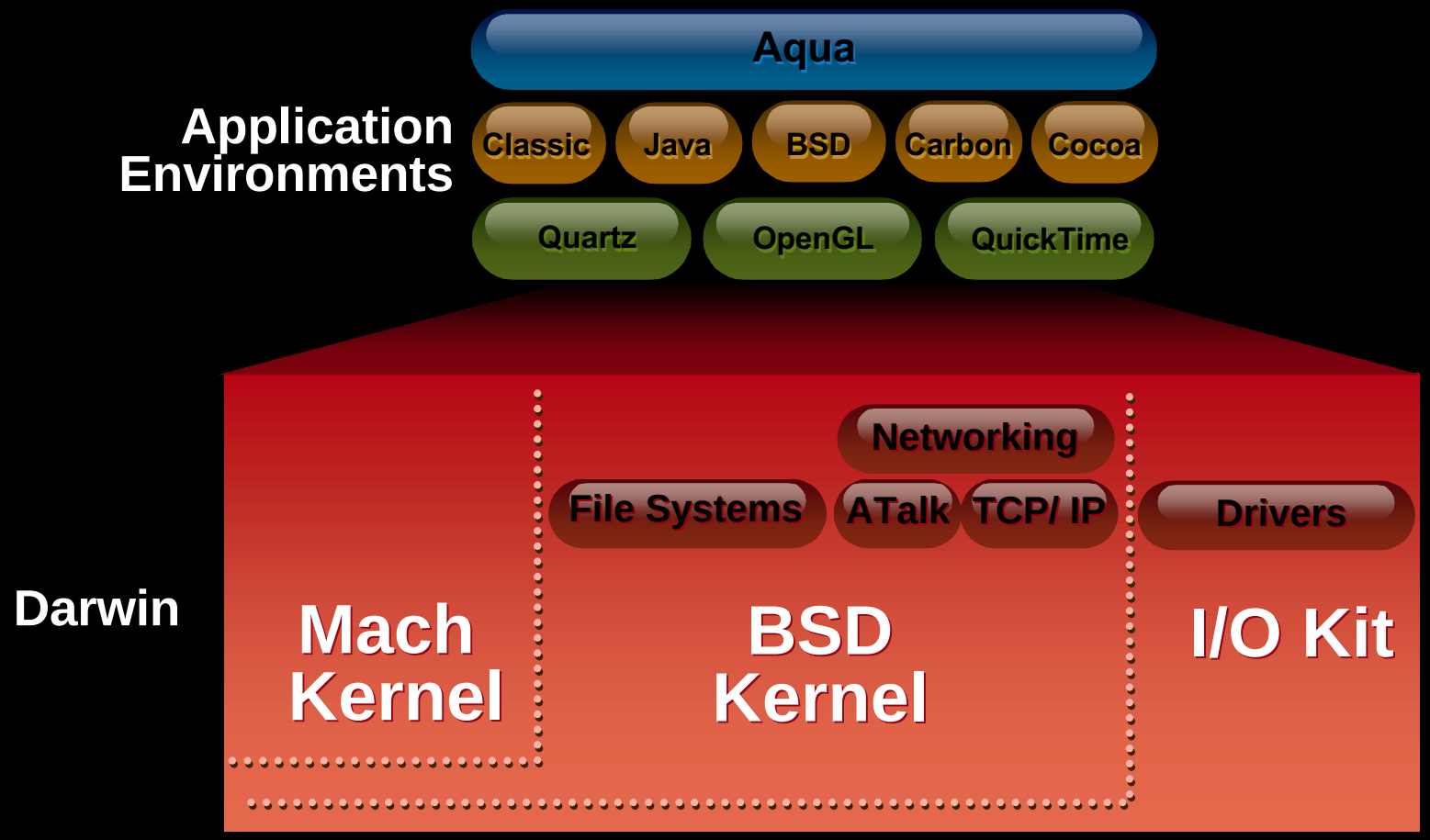

Aqua
Classic
Java
BSD
Carbon
Cocoa
Quartz
OpenGL
QuickTime
Darwin

Mac OS X Native Applications

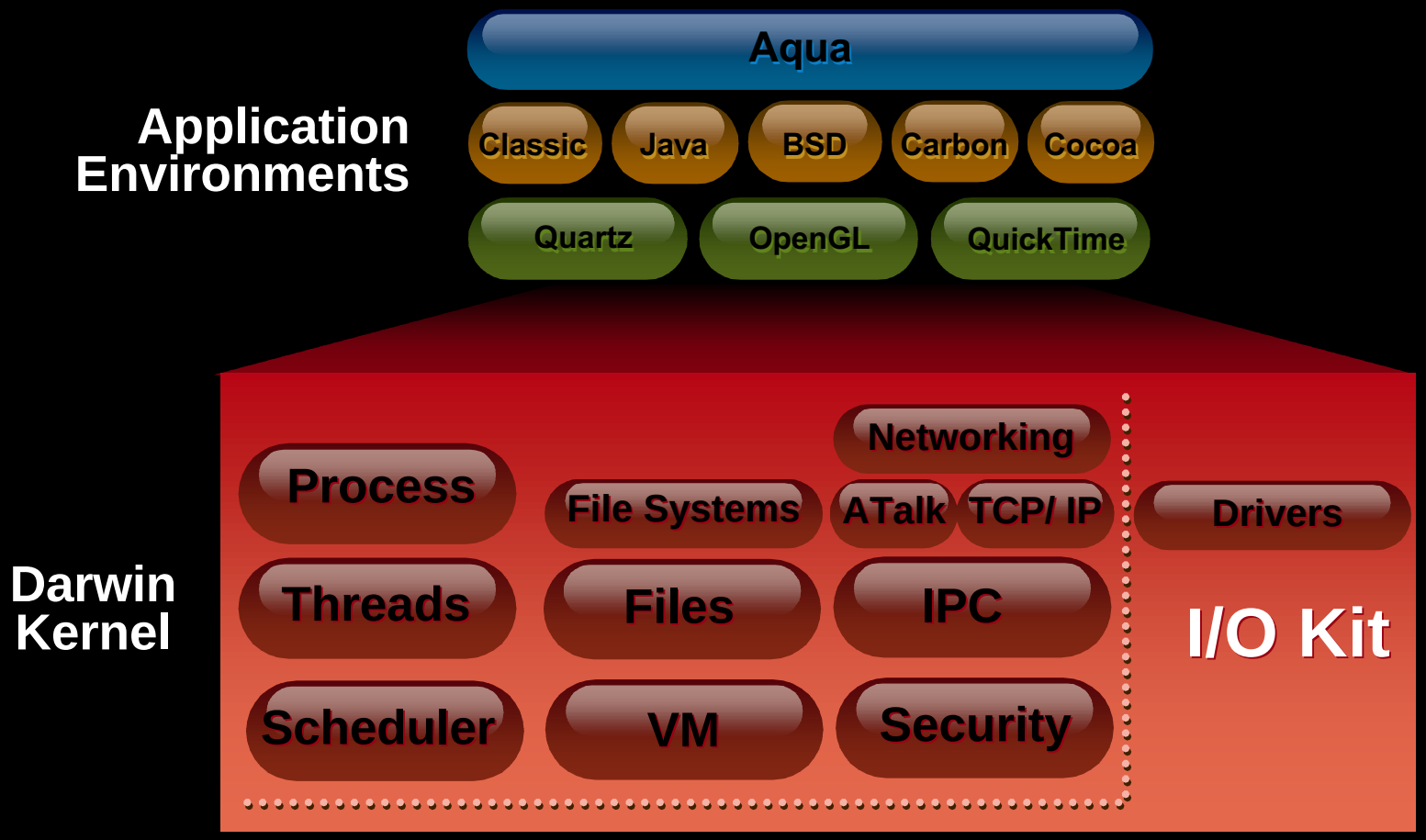
Some general truths about kernel resources

- Native applications are BSD processes
- BSD processes contain a Mach task
- APIs available "everywhere"
  - Avoid using lower-level APIs when possible
  - Exercise extreme care where unavoidable

Darwin Kernel



Darwin Kernel Services

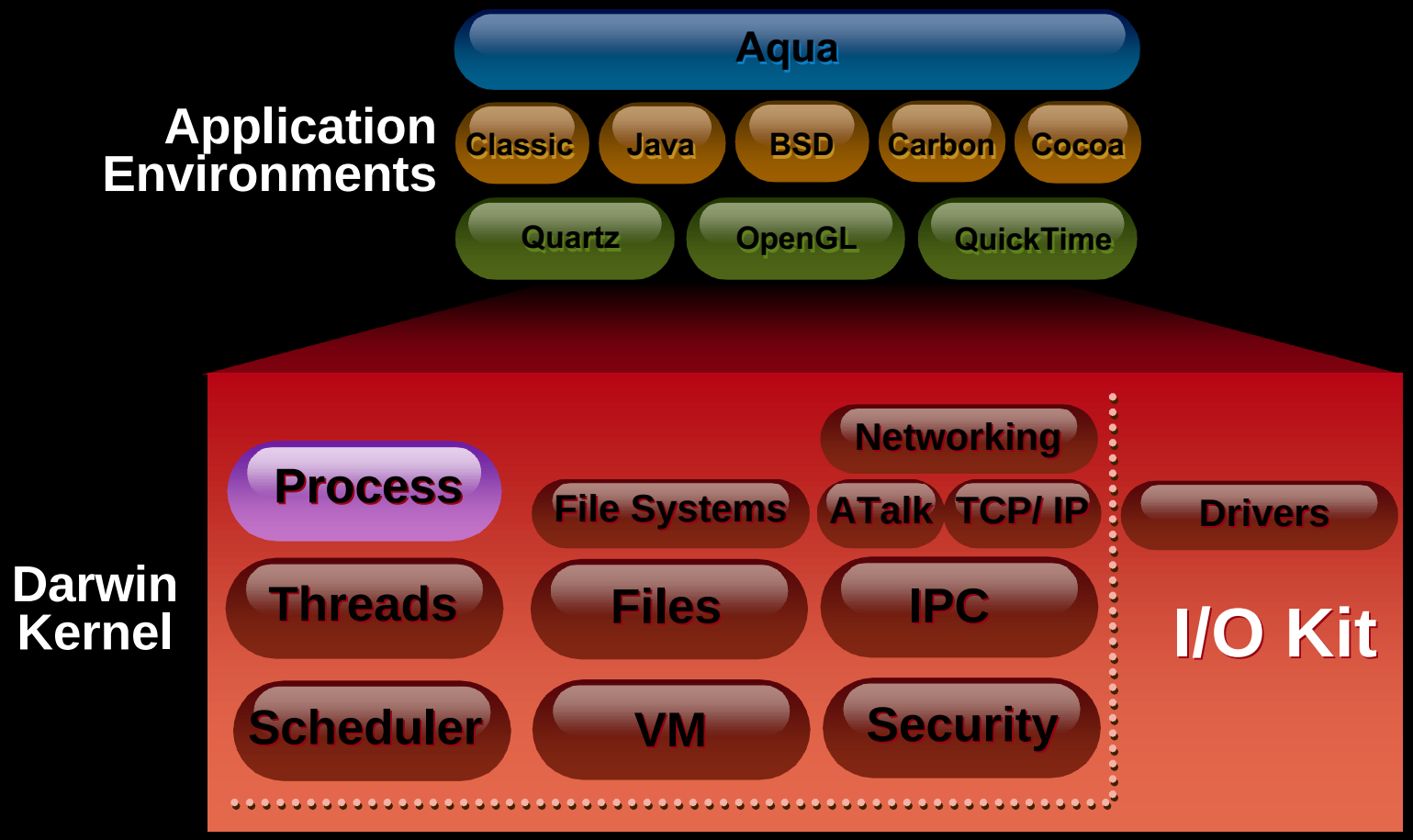


Darwin Kernel Origins

A harmony of technologies

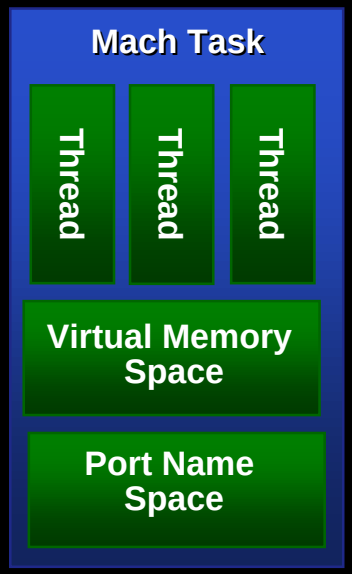
- Mach 3.0 (OSFMK73)
  - Virtual Memory, Scheduling, Inter-Process-Communication (IPC)
- 4.4BSD Lites 2
  - Process model, security, file access
- FreeBSD 3.0/3.1/3.2
  - NFS, Networking

Darwin Process Services



Mach Task Services

Tasks are the unit of resource ownership in Mach

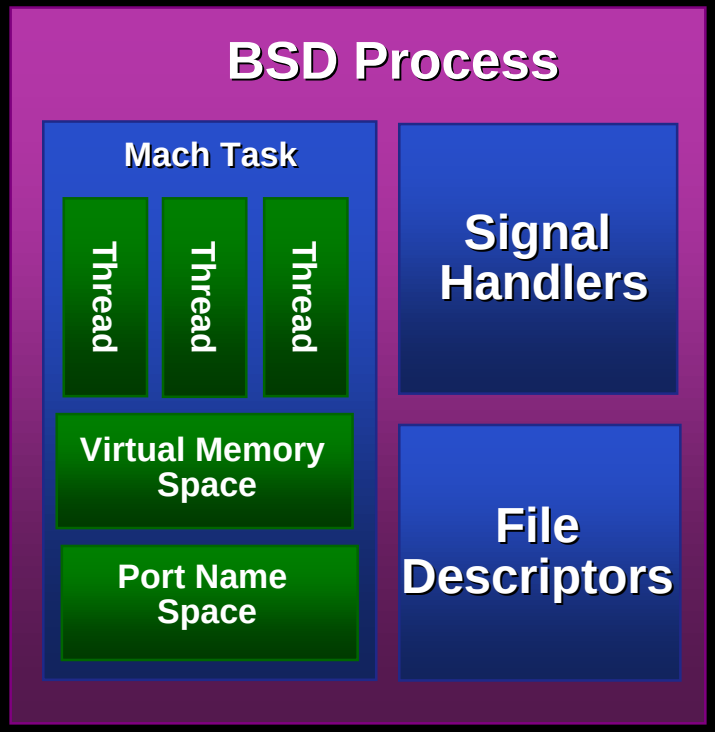


task_t

create, terminate
suspend, resume
info
threads
exception handlers
death notification

BSD Process Services

Add signal handling, parent-child relationships



PID

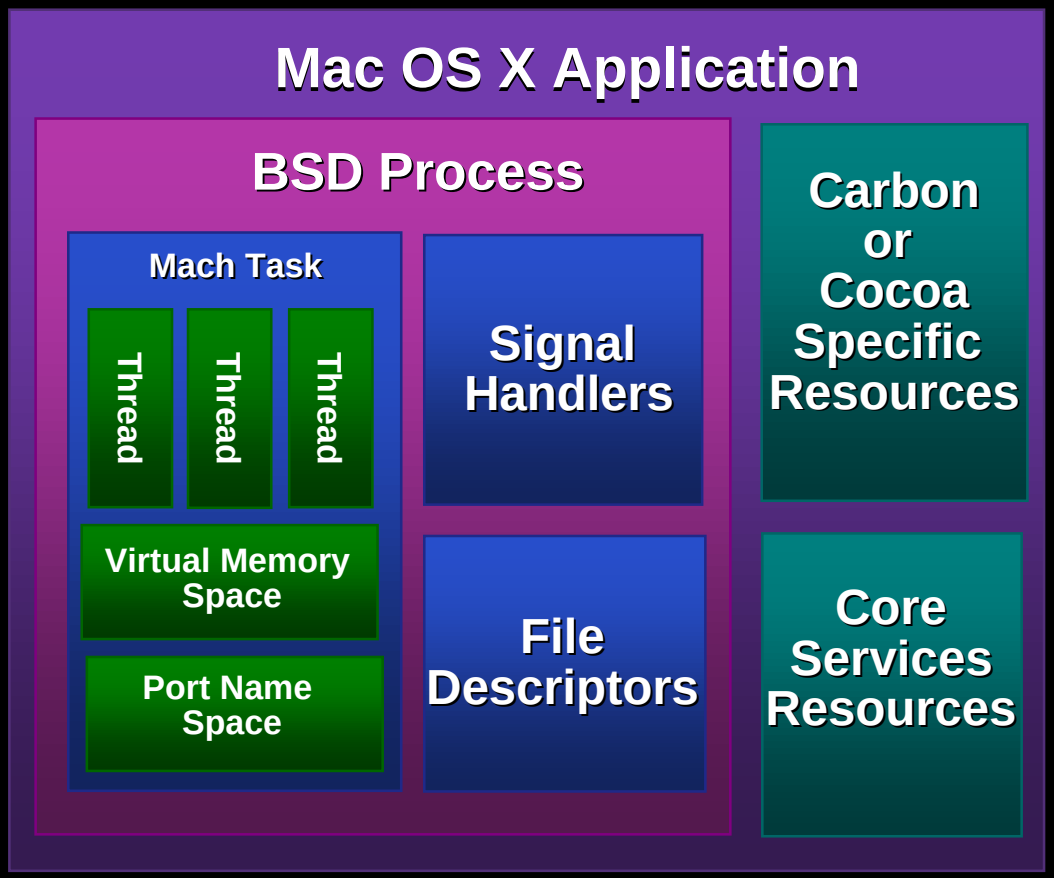
vfork, exec, exit
signal, kill
Ptrace
fork

mach_task_t

info, threads
exception handlers
death notification
others risky

Application Services

Add CPS management and core services resources



NSApplication Process Manager

CPS registration
resource management

PID

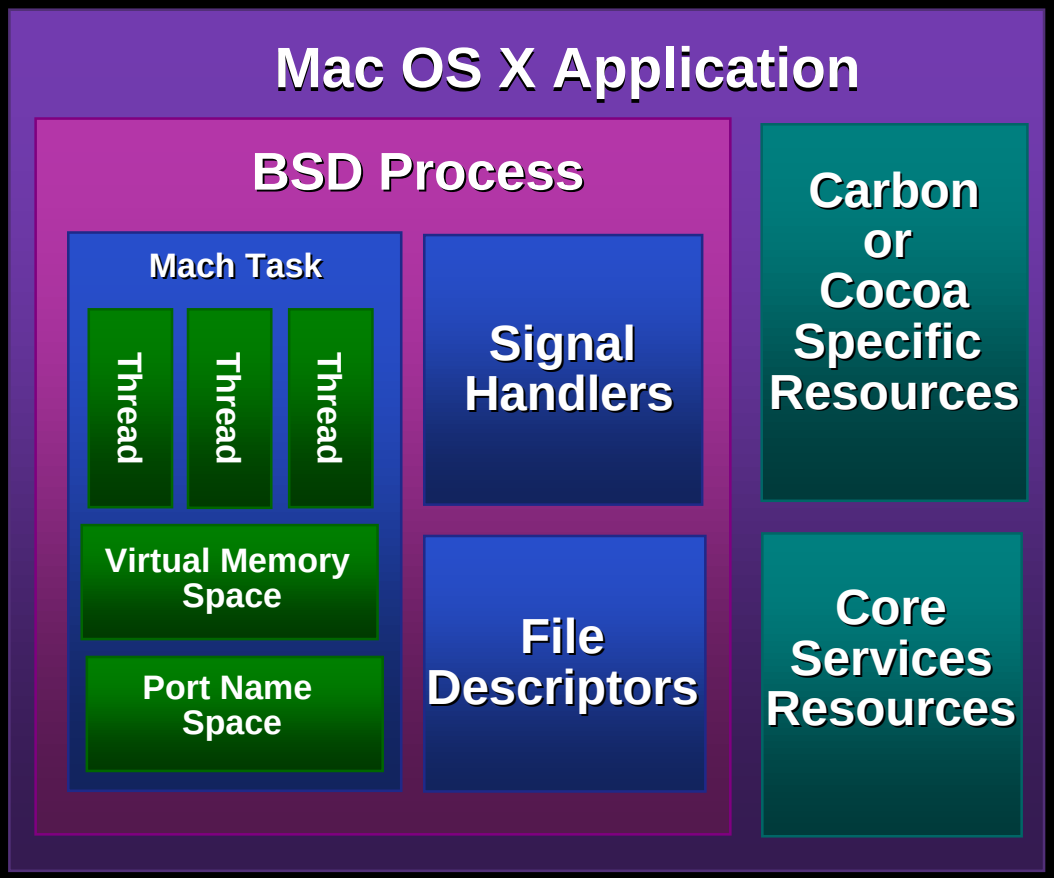
Most OK
fork without exec risky

mach_task_t

info, threads
exception handlers
death notification
others risky

Application Services

Tools and tips



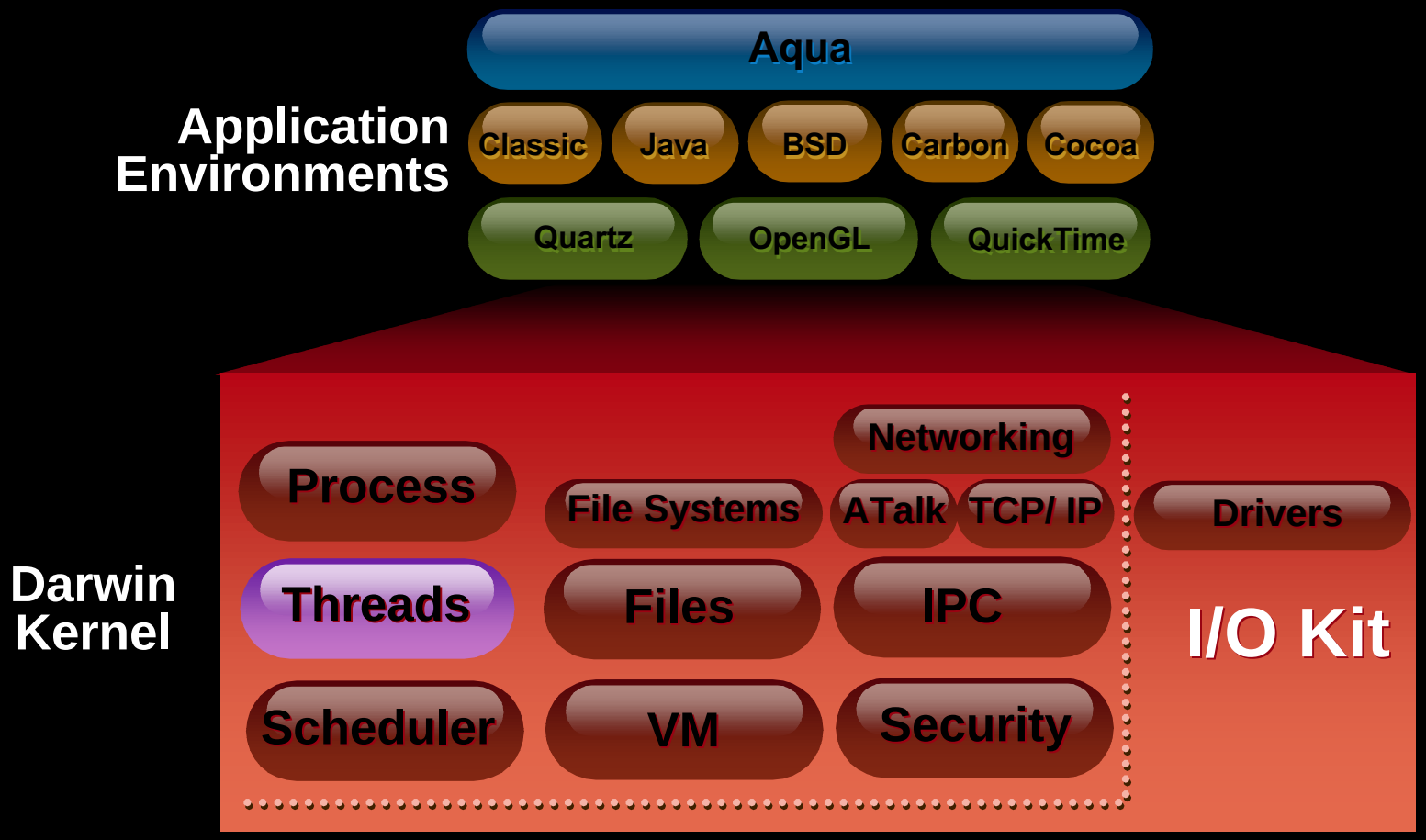
Tools

Dock, ForceQuit
ProcessViewer.app
top, ps
zprint

Tips

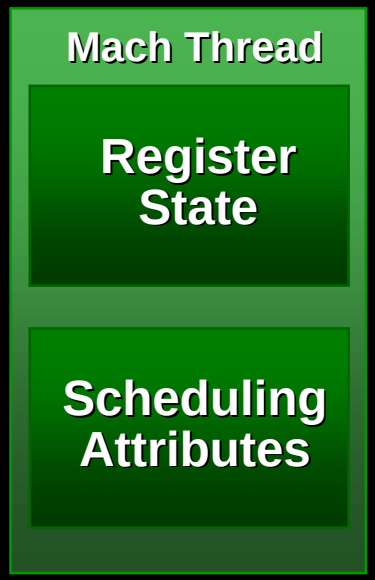
No mach-level resource limits are in place at the moment.
Leaky applications may panic the system

Darwin Thread Services



Mach Thread Services

Mach threads own no user-level resources

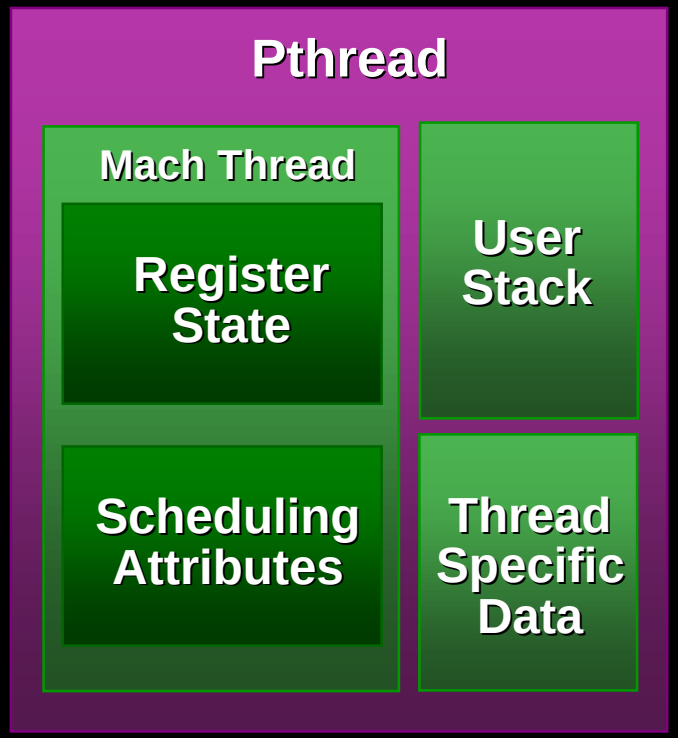


thread_t

create, terminate
suspend, resume
get_state, set_state
info
exception handlers
death notification
scheduling APIs

BSD Pthread Services

Pthreads define common resources of threads



pthread_t

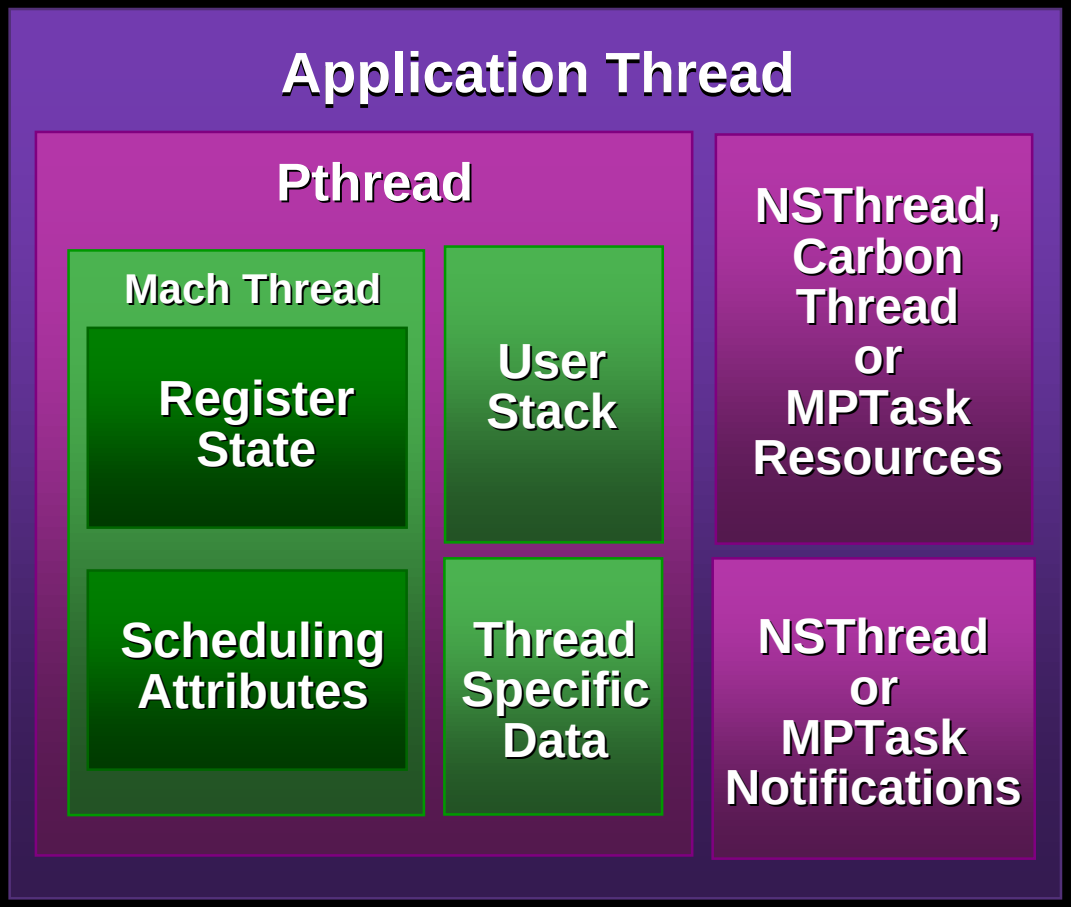
create, detach, join, exit
cancel
key_create, setspecific, getspecific

thread_t

info, threads
exception handlers
death notification
others risky

Application Thread Services

Give threads legacy semantics



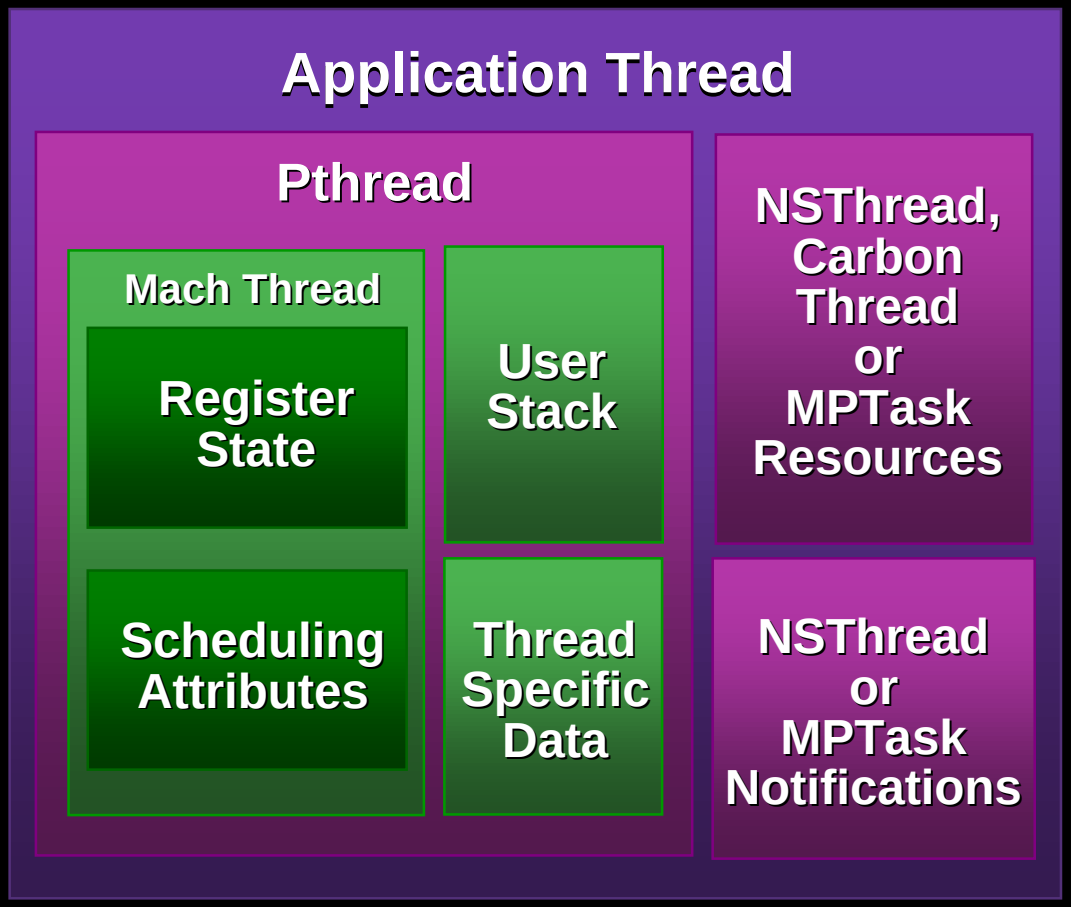
NSThread
MPTask
Thread Manager
tie to core services
tie to events

pthread_t
Most OK

thread_t
info, getstate, setstate
exception handlers
death notification
others risky

Application Thread Services

Tools and tips



Tools

ThreadViewer.app
Sample.app
top, ps

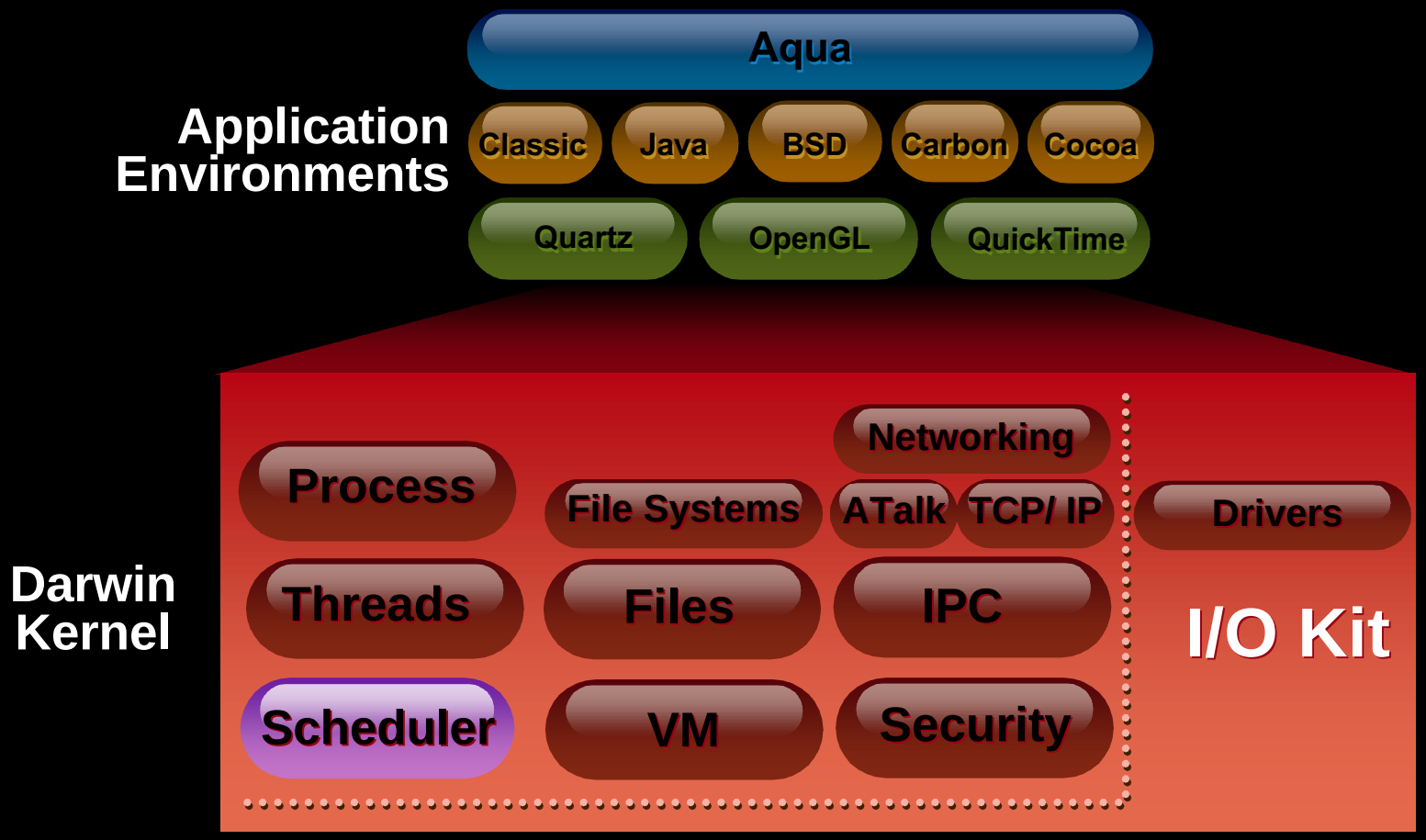
Tips

pthread_once for lazy init

Thread-level exception catching is least likely to conflict with others

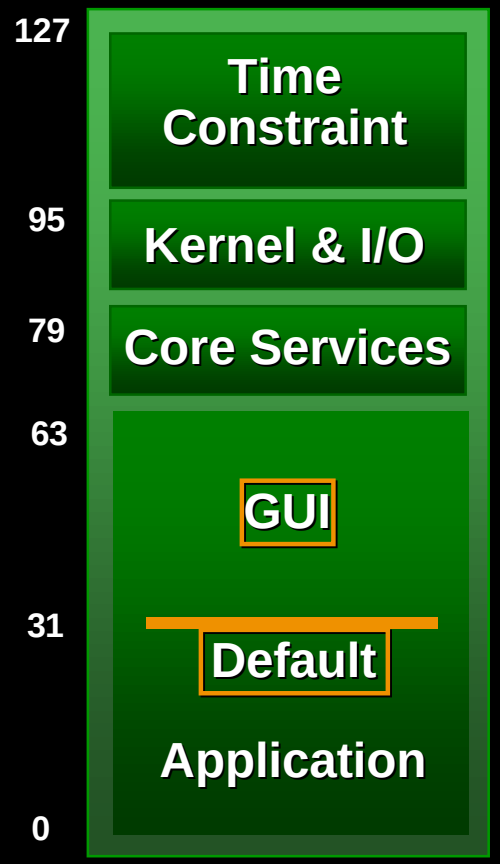
don't read TBR directly

Darwin Scheduler



Darwin Scheduler

Priority bands and policies (subject to change)



THREAD_TIME_CONSTRAINT_POLICY

you specify the constraints
kernel picks the priority
fixed priority - subject to limits
time constraint band

THREAD_STANDARD_POLICY

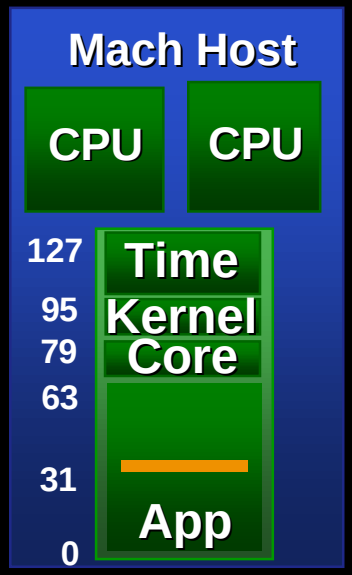
priorities adjust downward
based upon usage and contention

THREAD_PRECEDENCE_POLICY

relative weighting within task
orthogonal to other policy selection

Mach Scheduling Services

Multi-processor, fully preemptive (user and kernel)

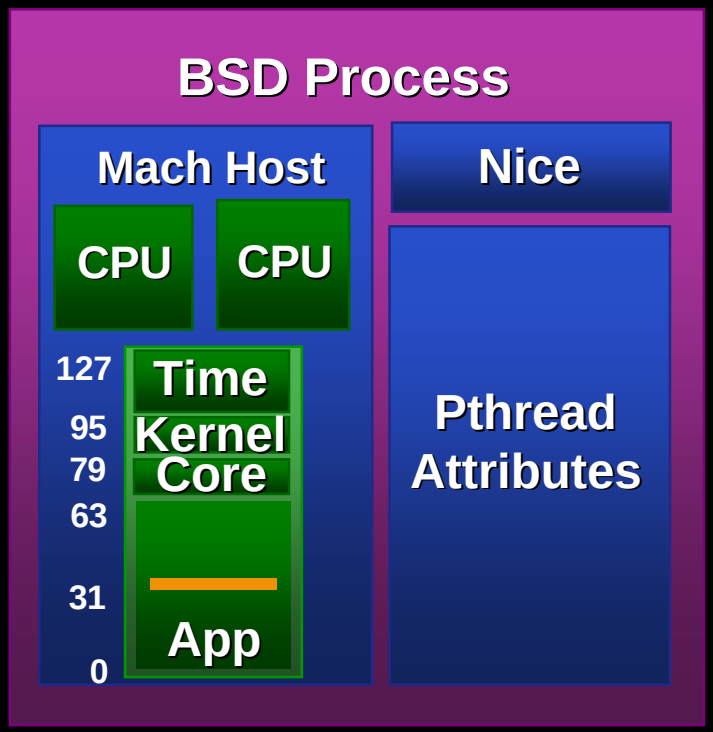


thread_t

policy_set, policy_get
suspend, resume
info
abort

BSD Scheduling Services

Rely heavily on Mach



PID

nice

not connected yet

pthread_t

setschedparam, getschedparam

minimal support as of now

thread_t

policy_set, policy_get

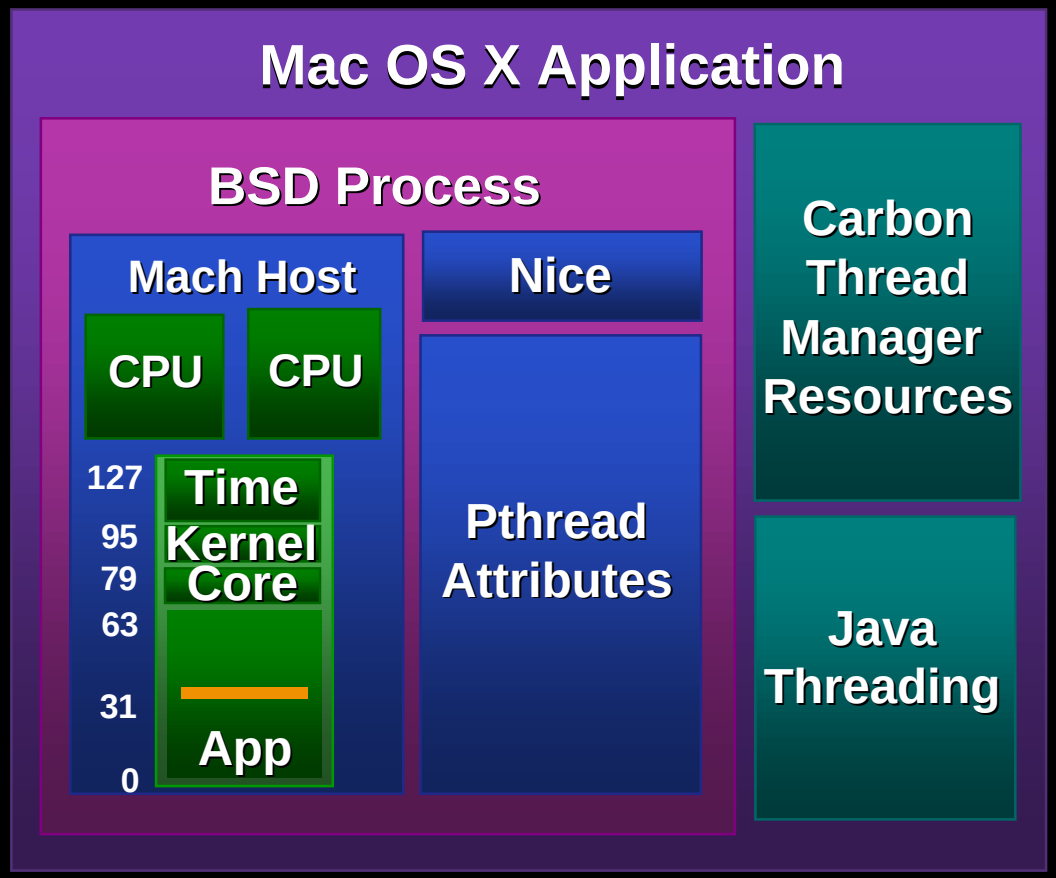
suspend, resume

info

abort

Application Scheduling Services

Sophistication beyond the core



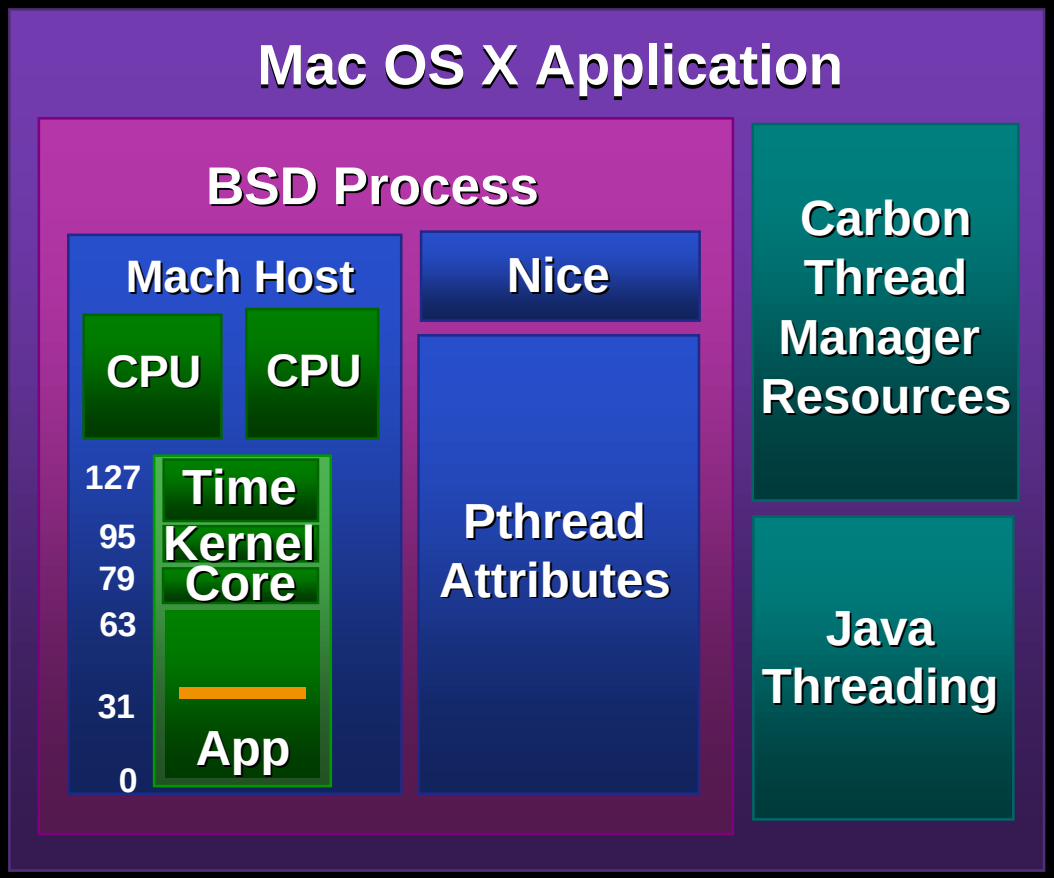
NSThread
Thread dictionaries

ThreadManager
deferred threads
cooperative threading
one Mach thread per

pthread_t
thread_t
OK - with sane policies

Application Scheduling

Tools and tips



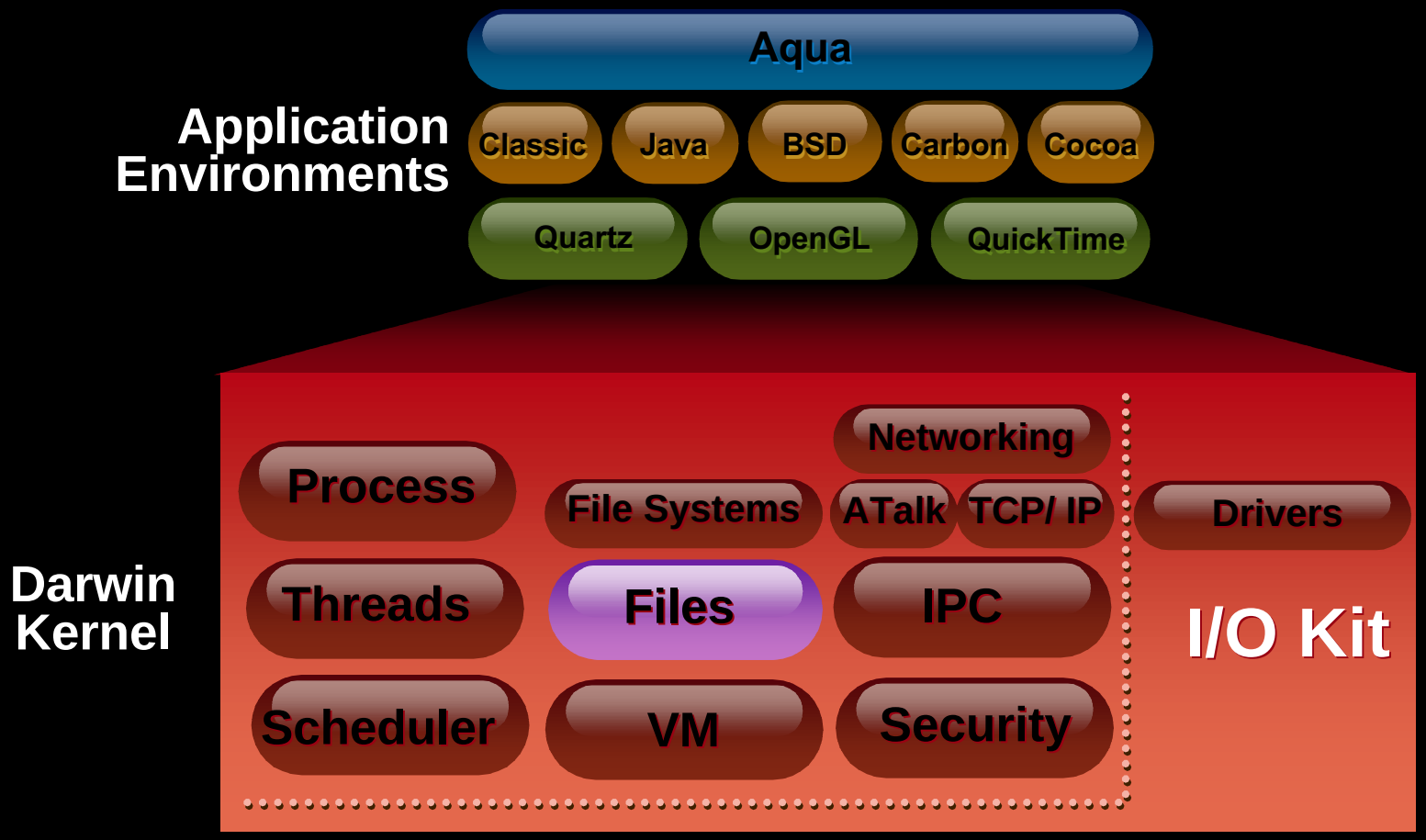
Tools

ThreadViewer.app
Sample.app
top, ps -M, sc_usage

Tips

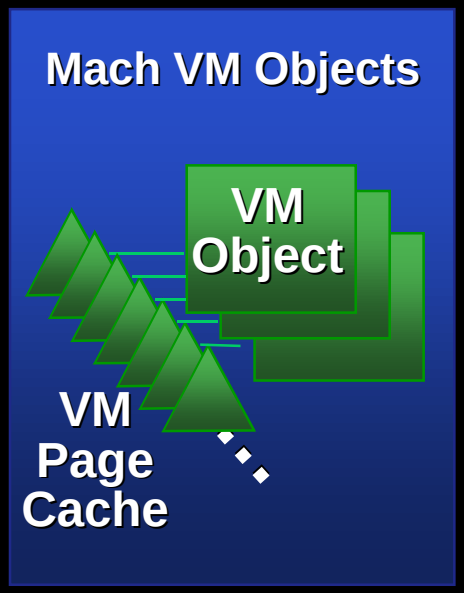
Do not use old Mach RR and TS policy APIs
they are deprecated

Darwin File Services



Mach VM Object Cache

Manages all cached data through VM object pages

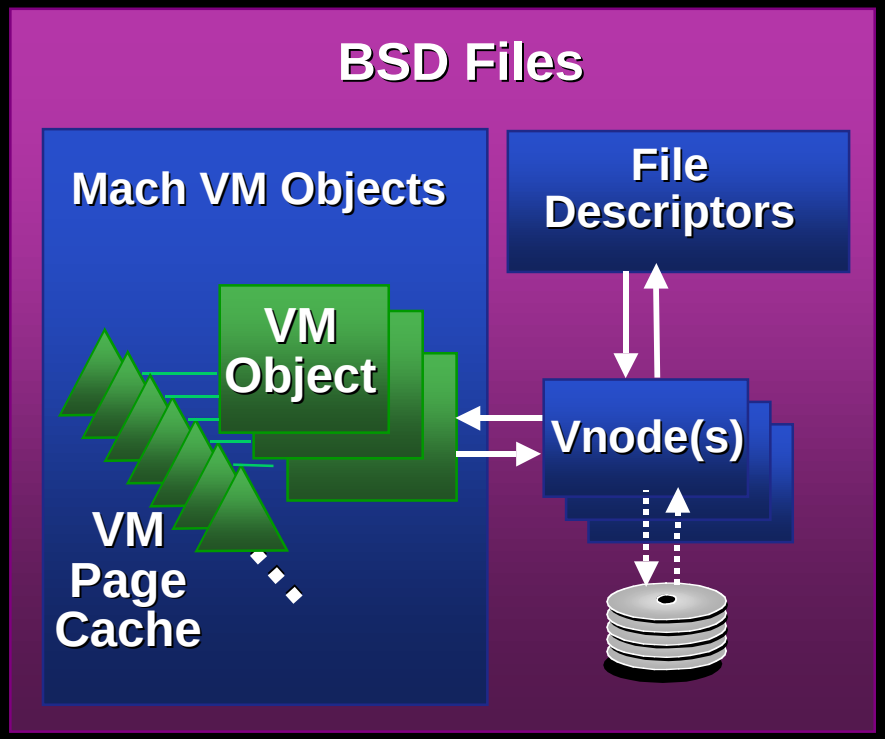


vm_object_t
vm_page_t

Internal Mach VM objects for managing virtual memory

BSD File Services

Control access to all files through descriptors



fd

File

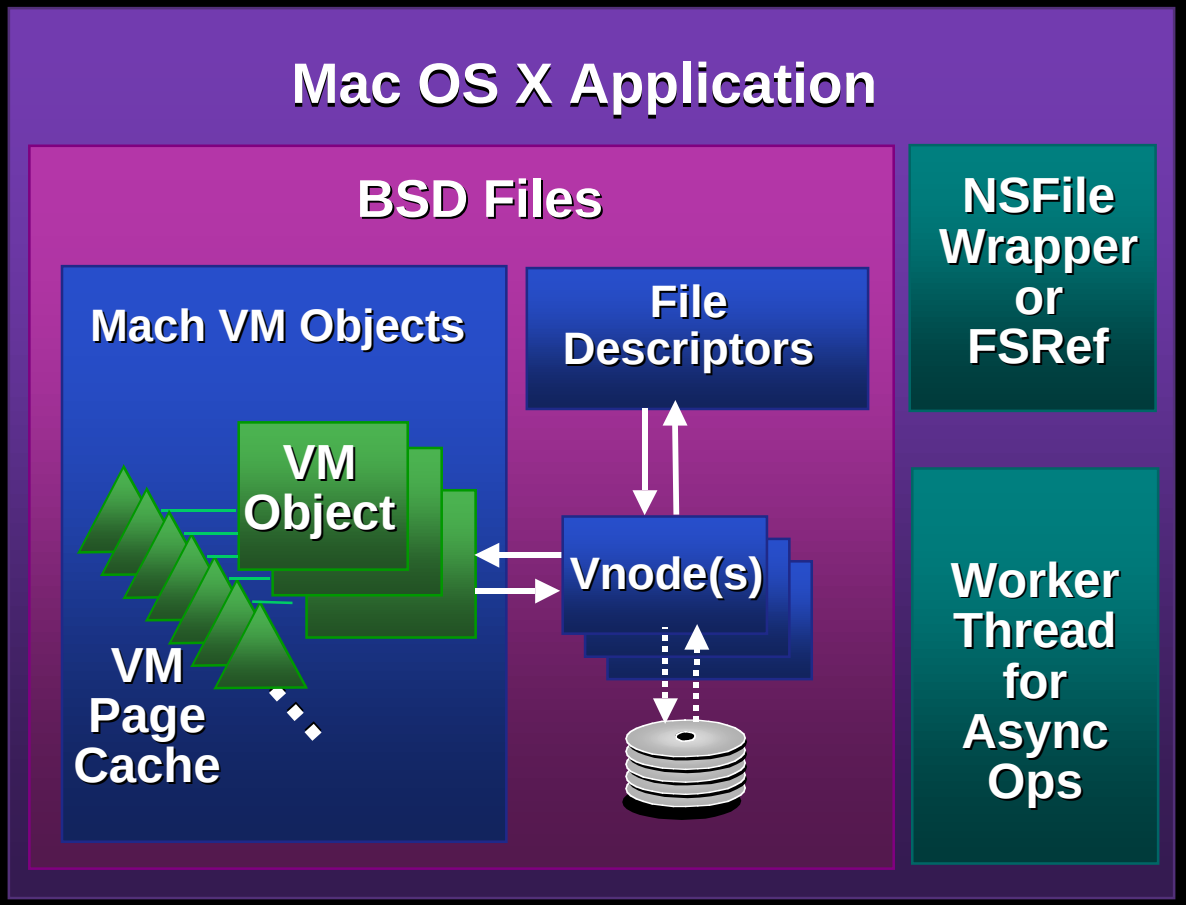
open close
read, write
mmap

Vnode

synchronizes file I/O with the VM object page cache

Application File Services

Extend BSD/POSIX semantics



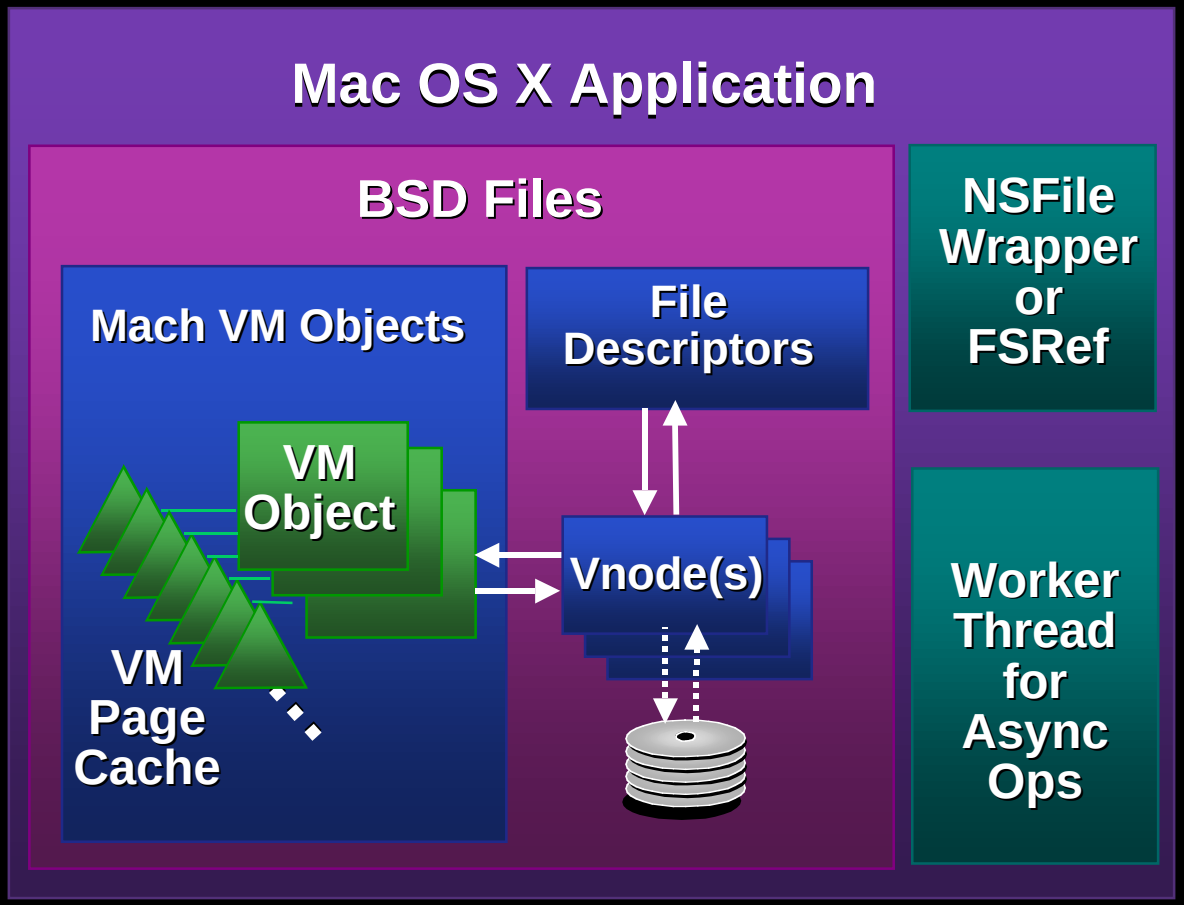
NSFileWrapper
FileManager
async ops
via worker thread

resource forks

fd
File
open close
read, write
mmap

File Services

Tools and tips



Tools

fs_usage
lsof

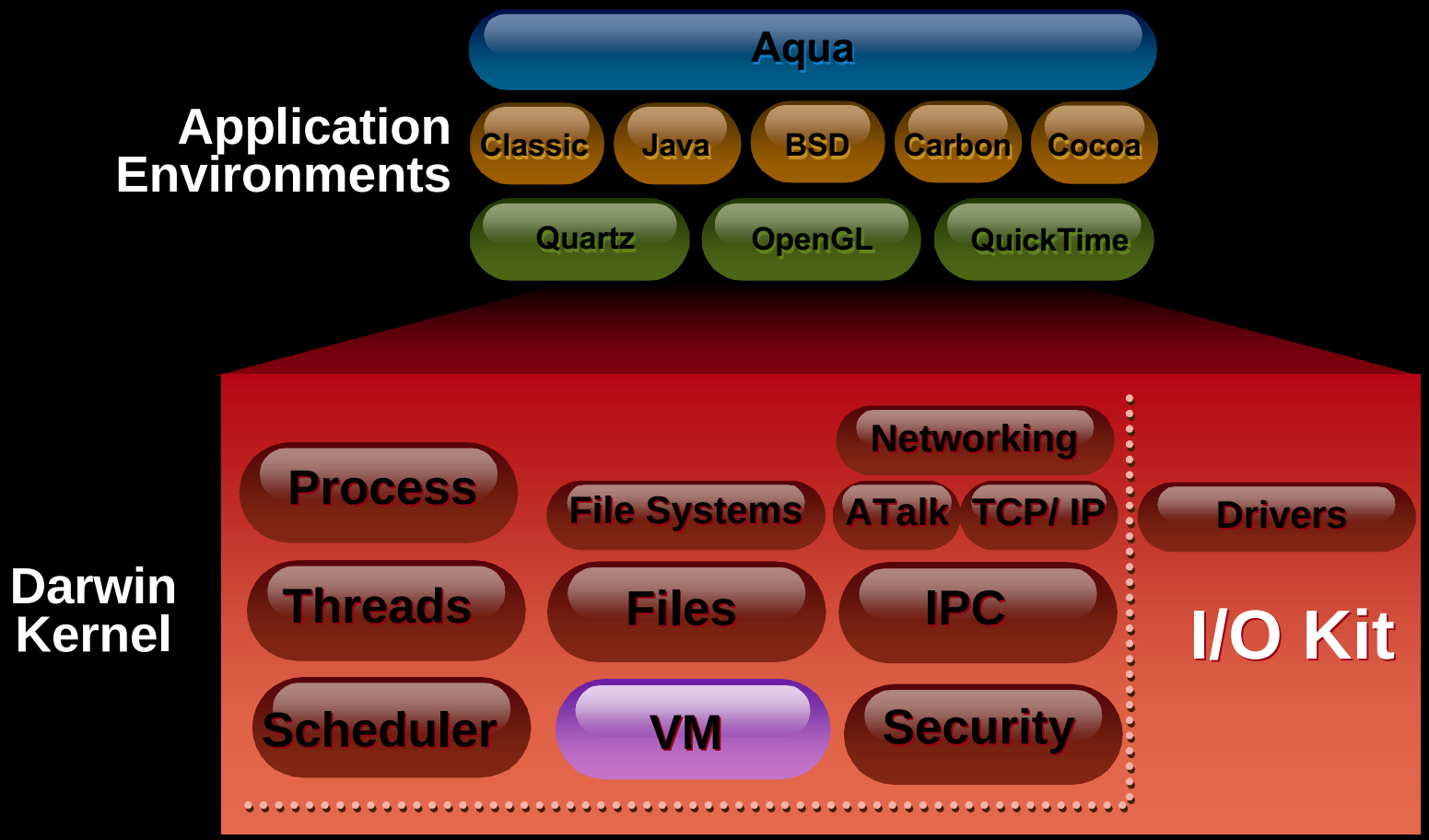
Tips

Use NOCACHE options

256 open file soft limit per process

system-wide tunable limit on vnodes

Darwin Virtual Memory

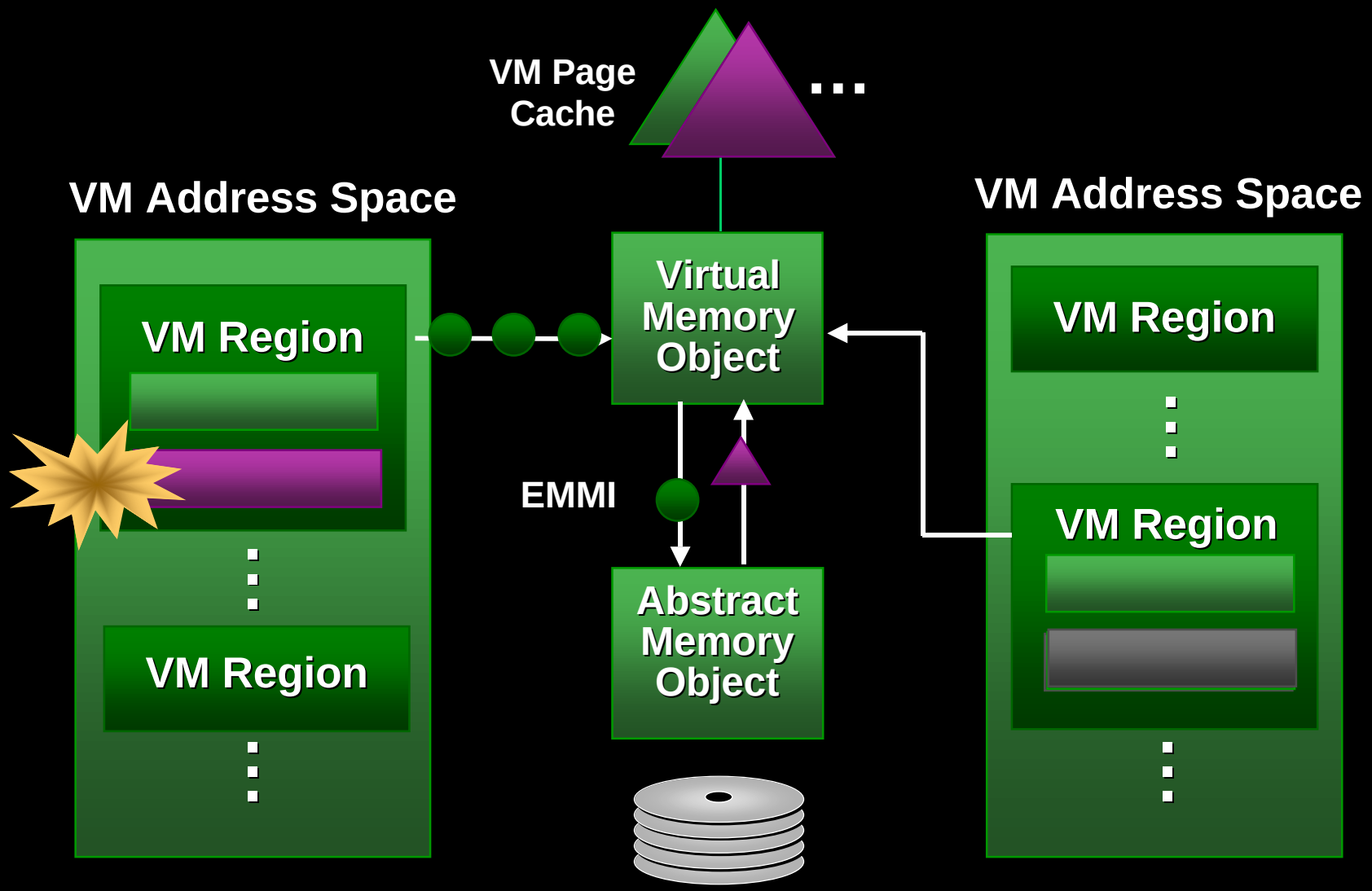


Virtual Memory

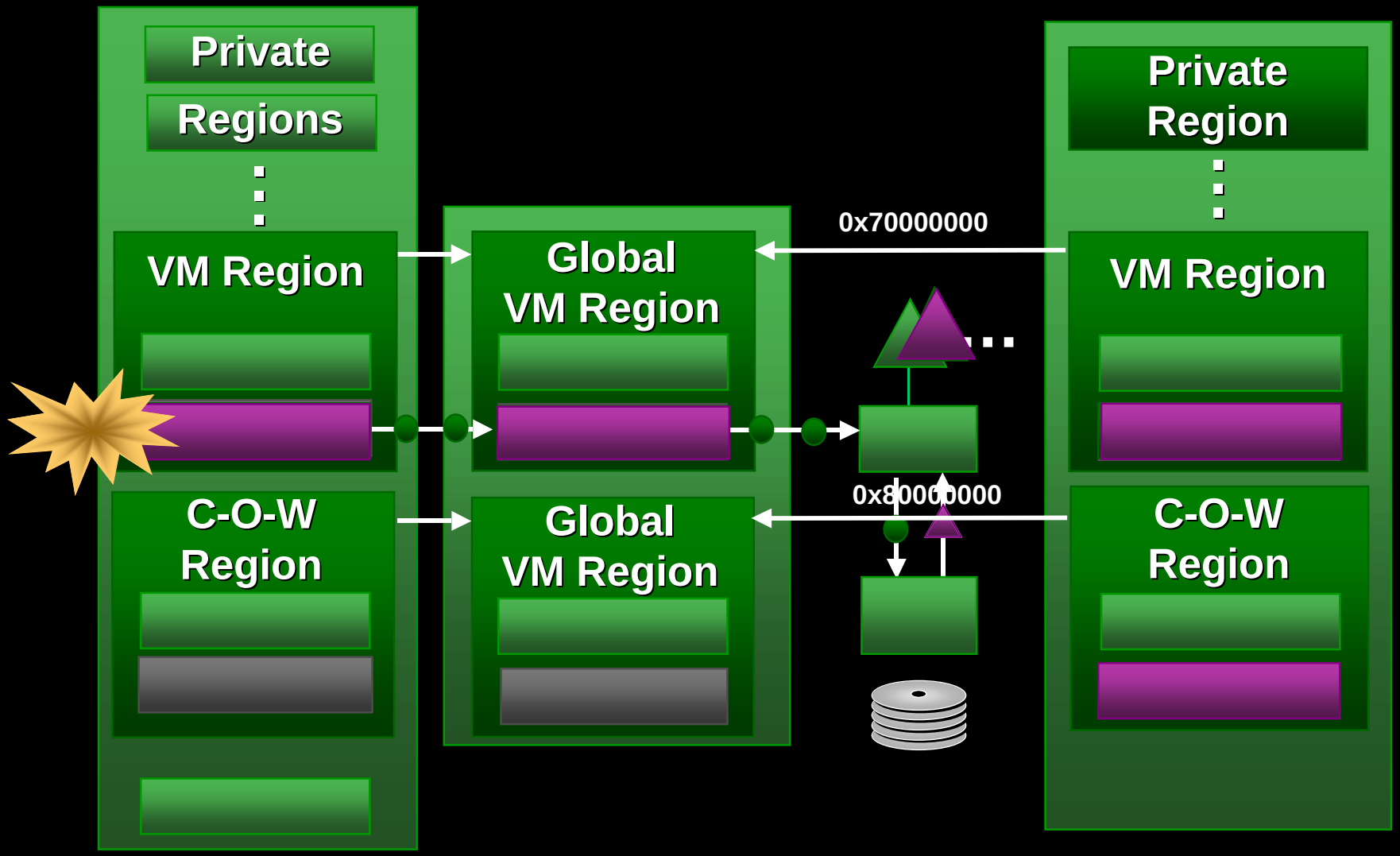
Mach controls most aspects of virtual memory

- Protected address spaces
- Layout flexibility
- Controlled sharing
- Copy-on-write and lazy evaluation

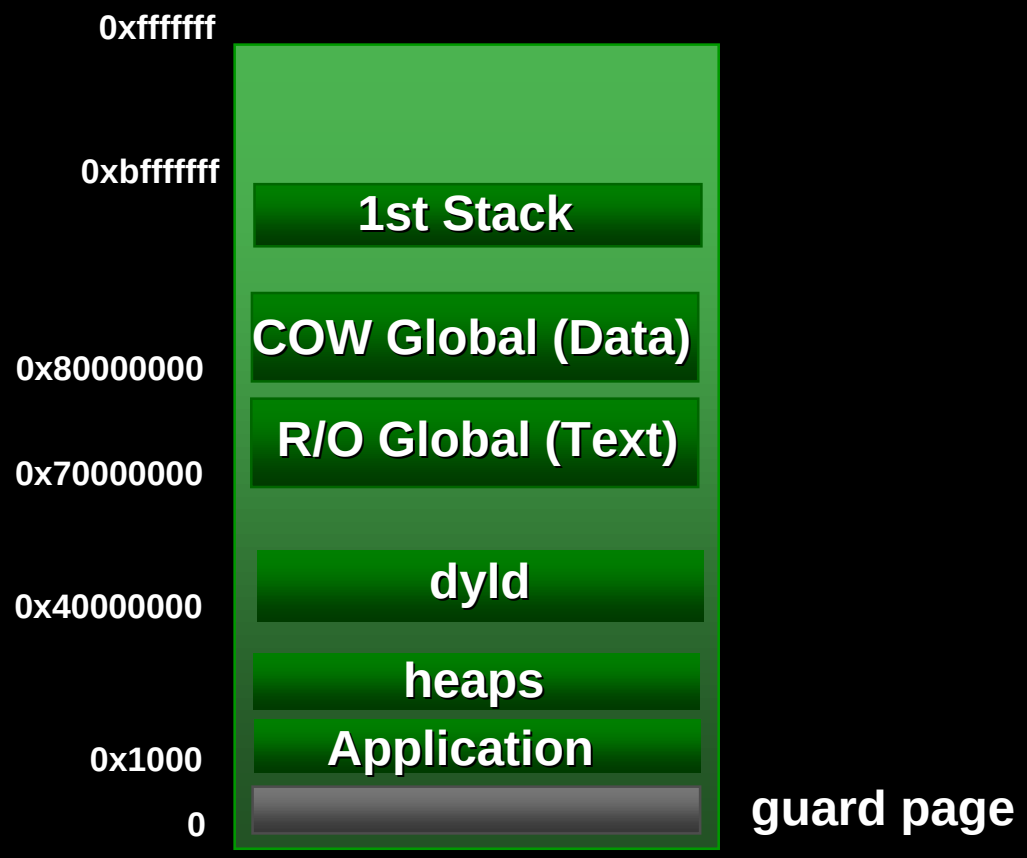
Virtual Memory



Shared Region Support

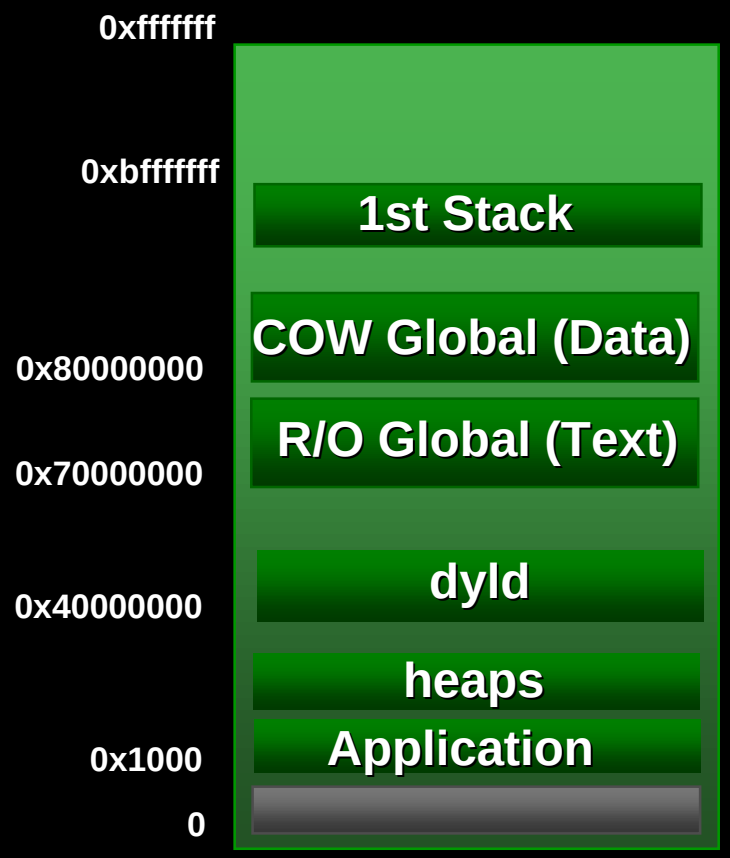


Typical Mach-O Memory Layout



Virtual Memory Services

Tools and tips



Tools

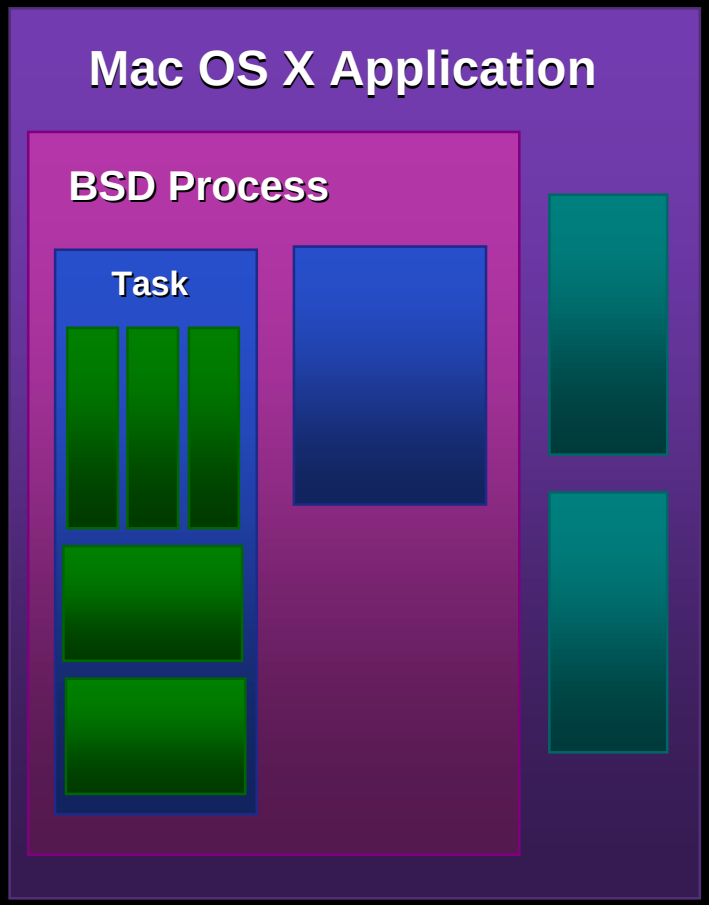
MallocDebug.app
vmmap
top

Tips

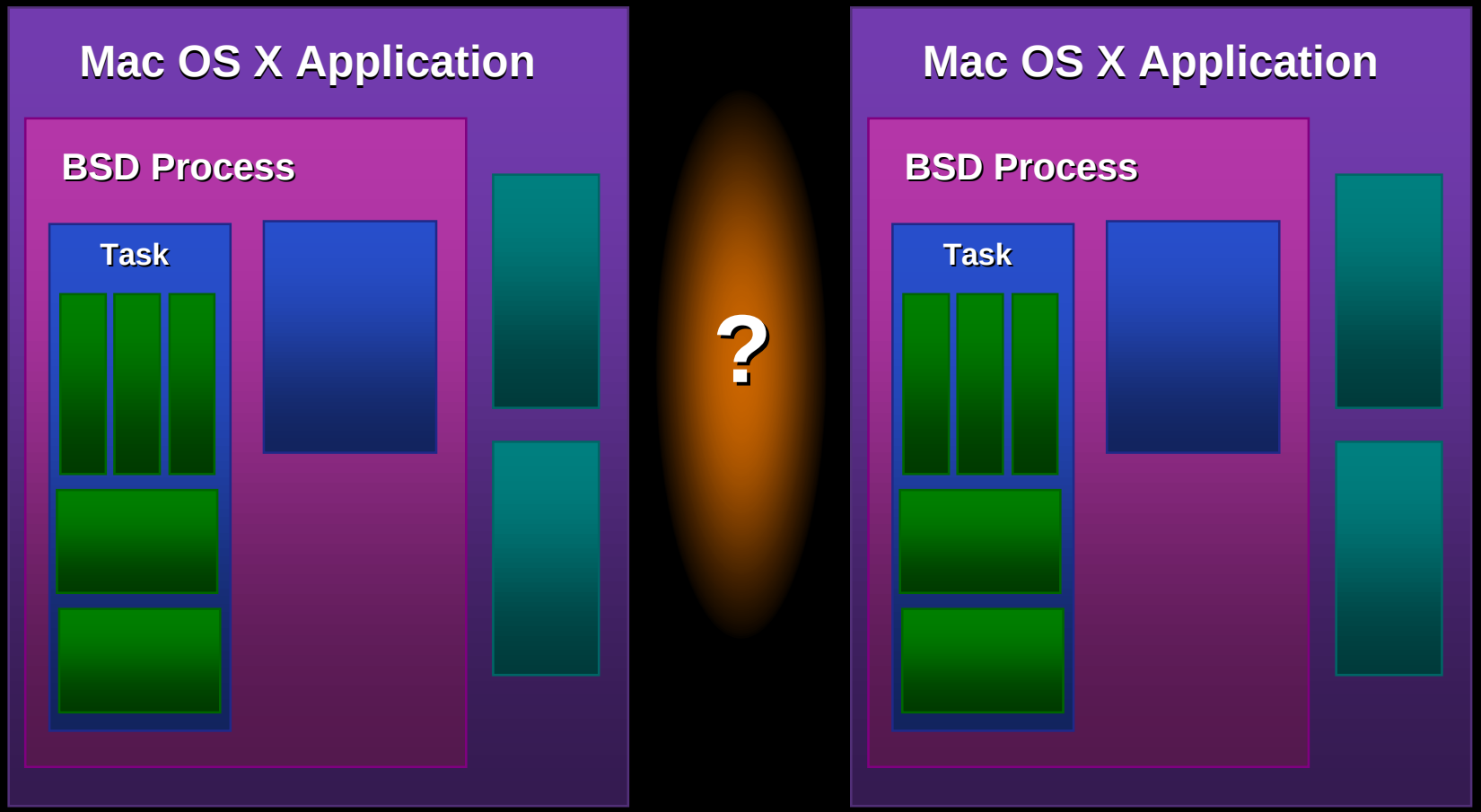
memory allocation defaults to first fit near zero

kill or deallocate dirty but not needed pages

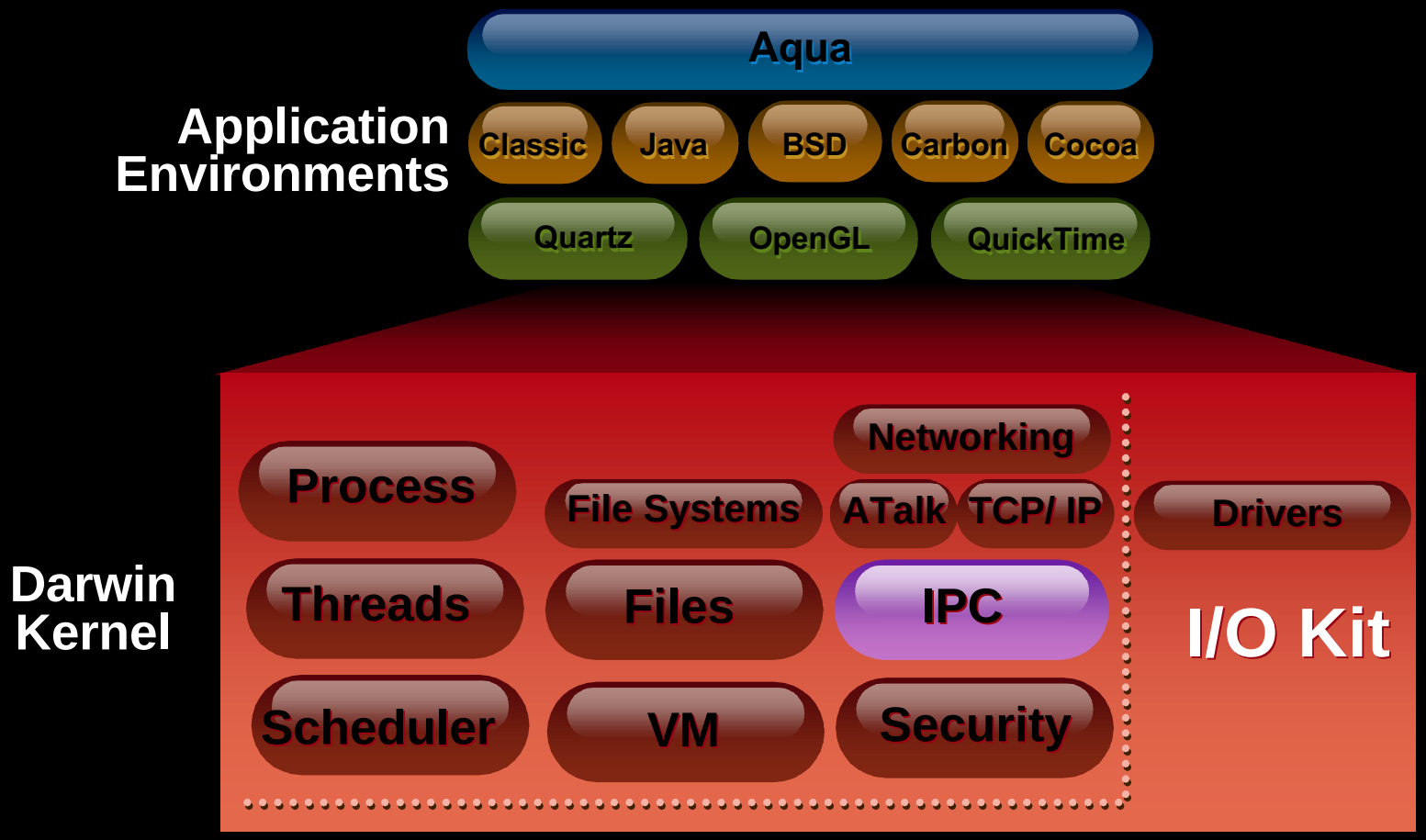
We Have Protection…



. . . What About Cooperation?

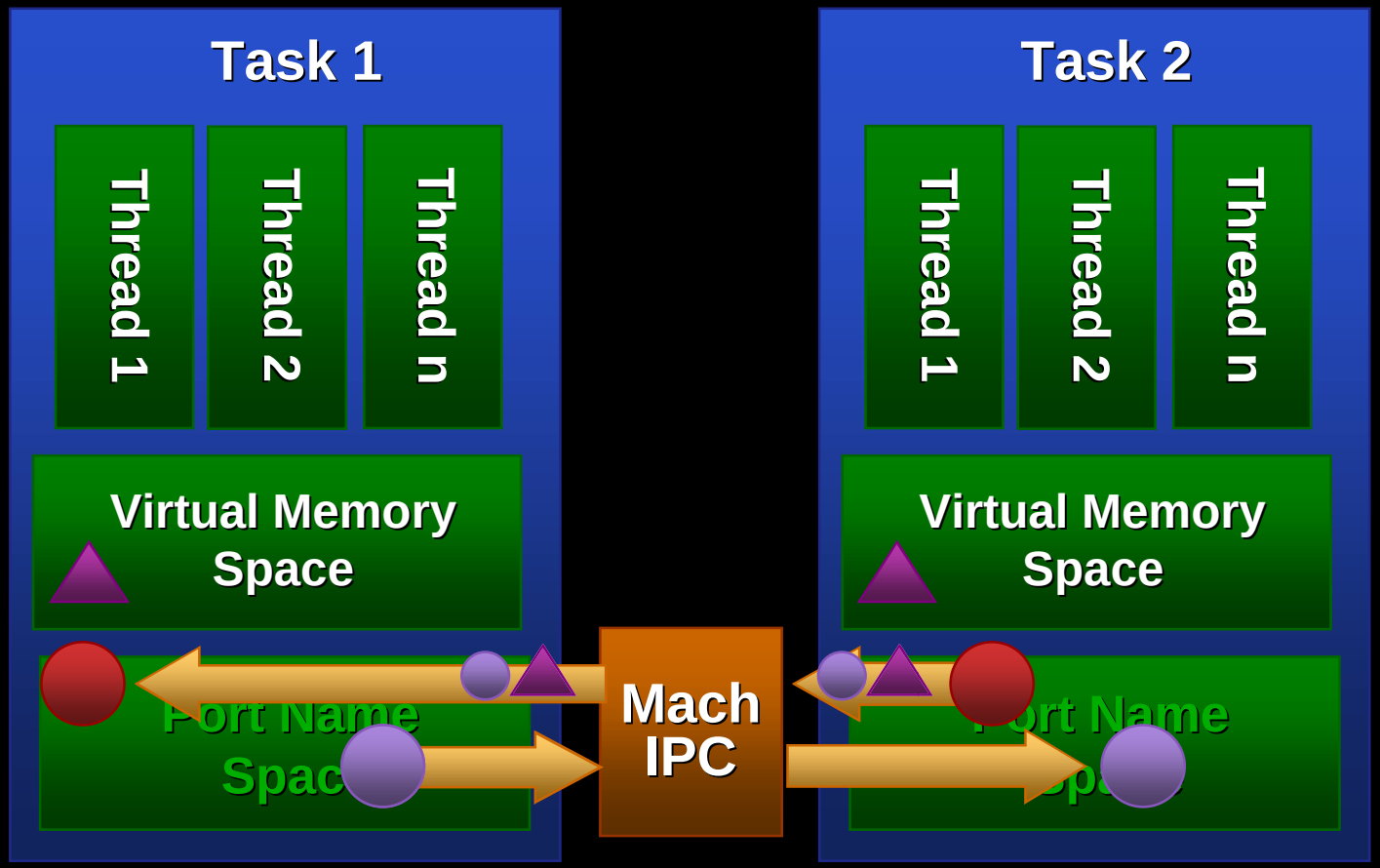


Darwin IPC Services



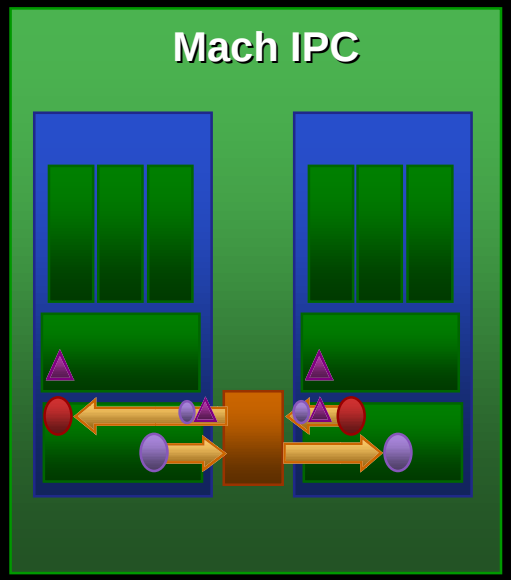
Mach IPC Basics

Can transfer memory and ports at the same time



Mach IPC Services

Fundamental types

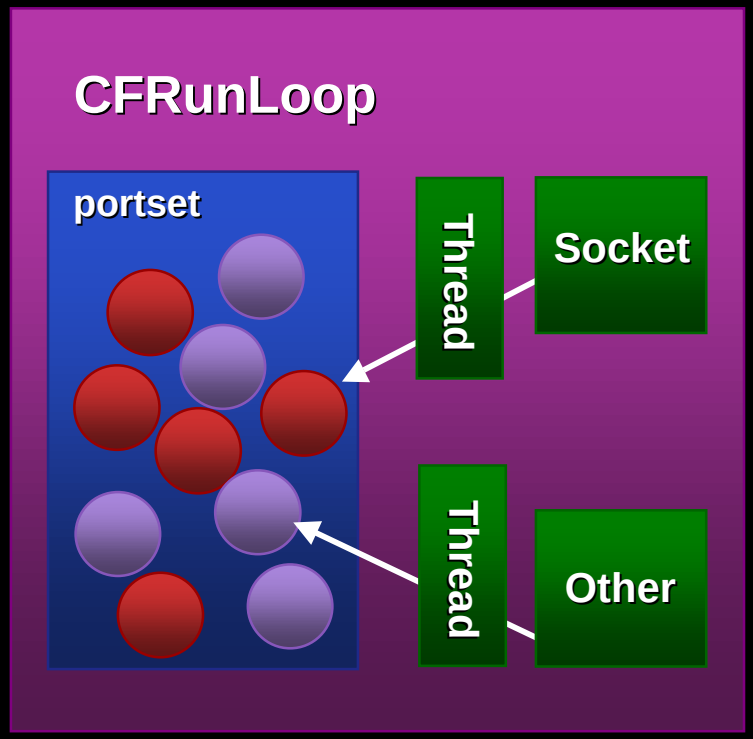


mach_port_t

communication endpoint
message queue
semaphore
lockset
port set

Mach IPC and CFRunLoop

Collection of port rights representing event sources



CFRunLoopRef

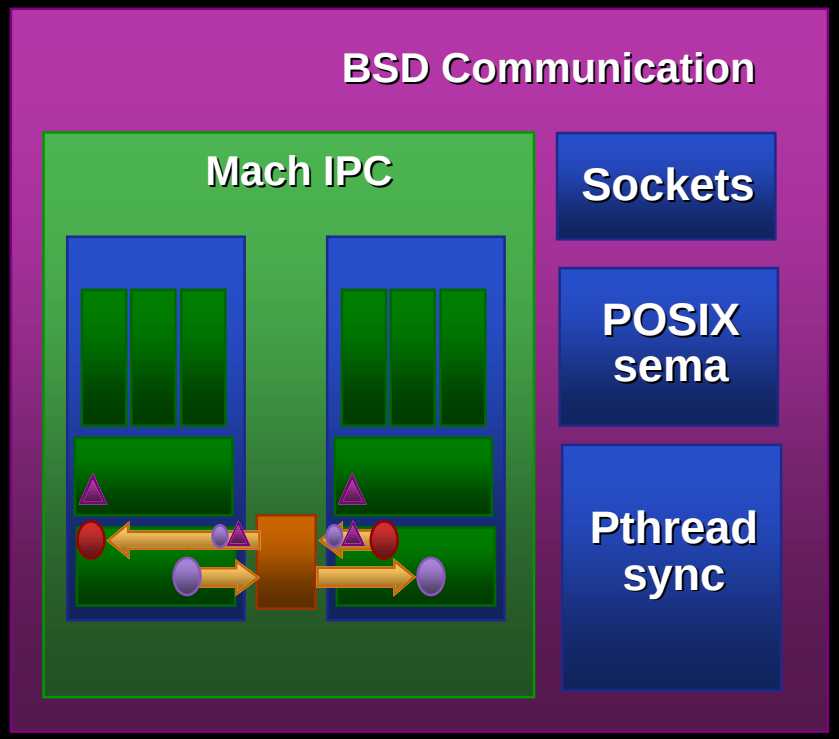
single wait point
diverse event sources
poll model
event driven model
worker threads

mach_port_t

communication endpoint
message queue
semaphore
lockset
port set

BSD IPC Services

A collection of POSIX standards



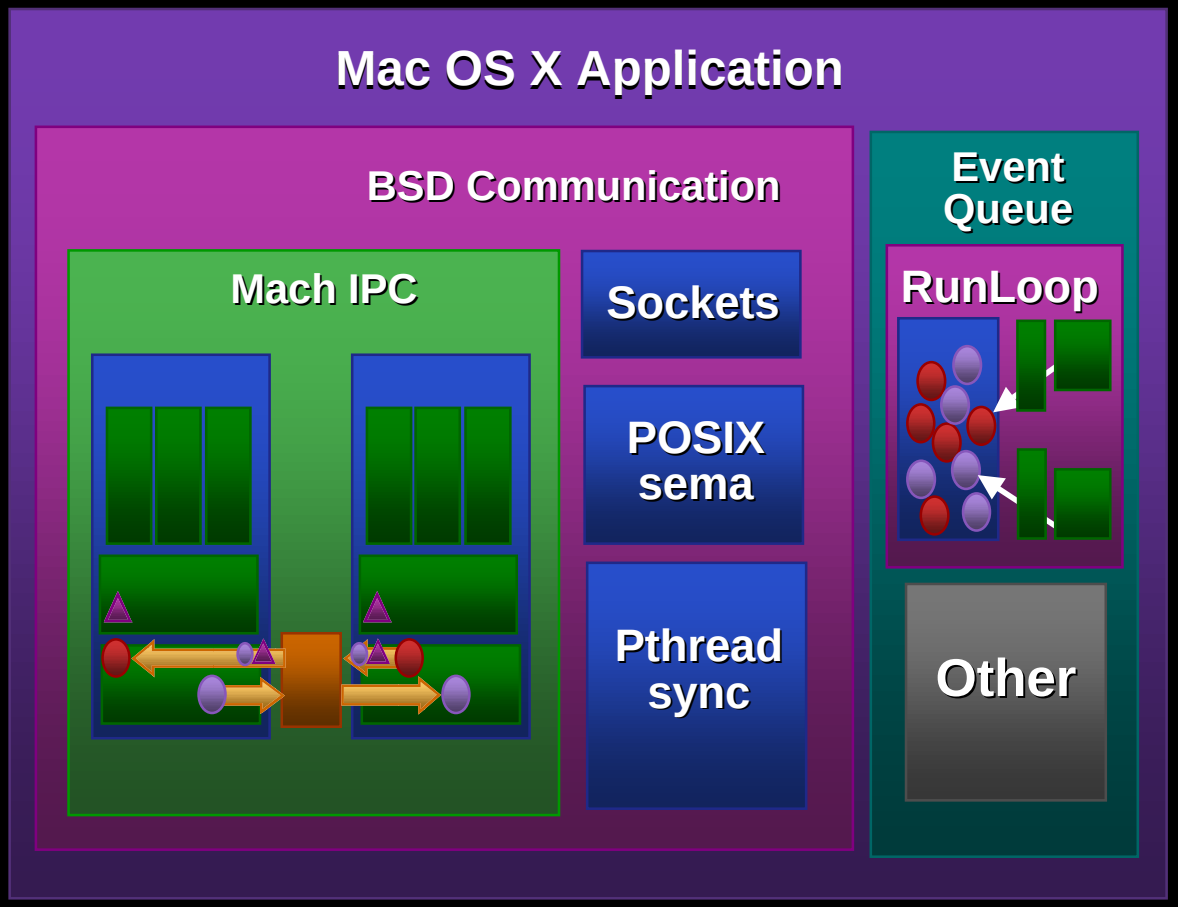
pthread_mutex_t
pthread_cond_t
only intra-process for now

sem_t
open, close, wait, post
named only for now

CFRunLoopRef
mach_port_t
supported through System framework and CFLite

Application IPC Services

The intersection of it all

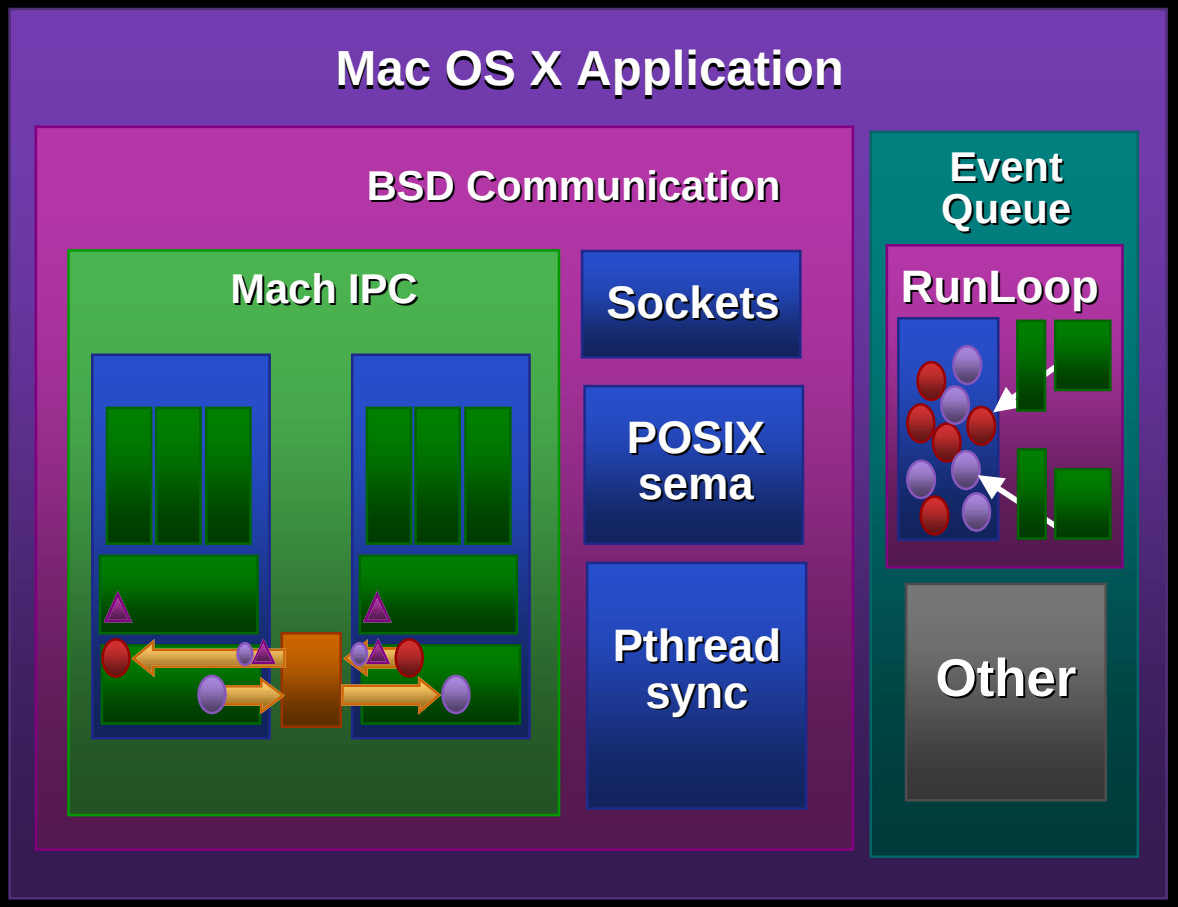


AppleEvents
the preferred low bandwidth method

NSEvent
EventRef
synthesis of multiple even t sources

Application IPC

Tools and tips



Tools

Sample.app
top, sc_usage
zprint

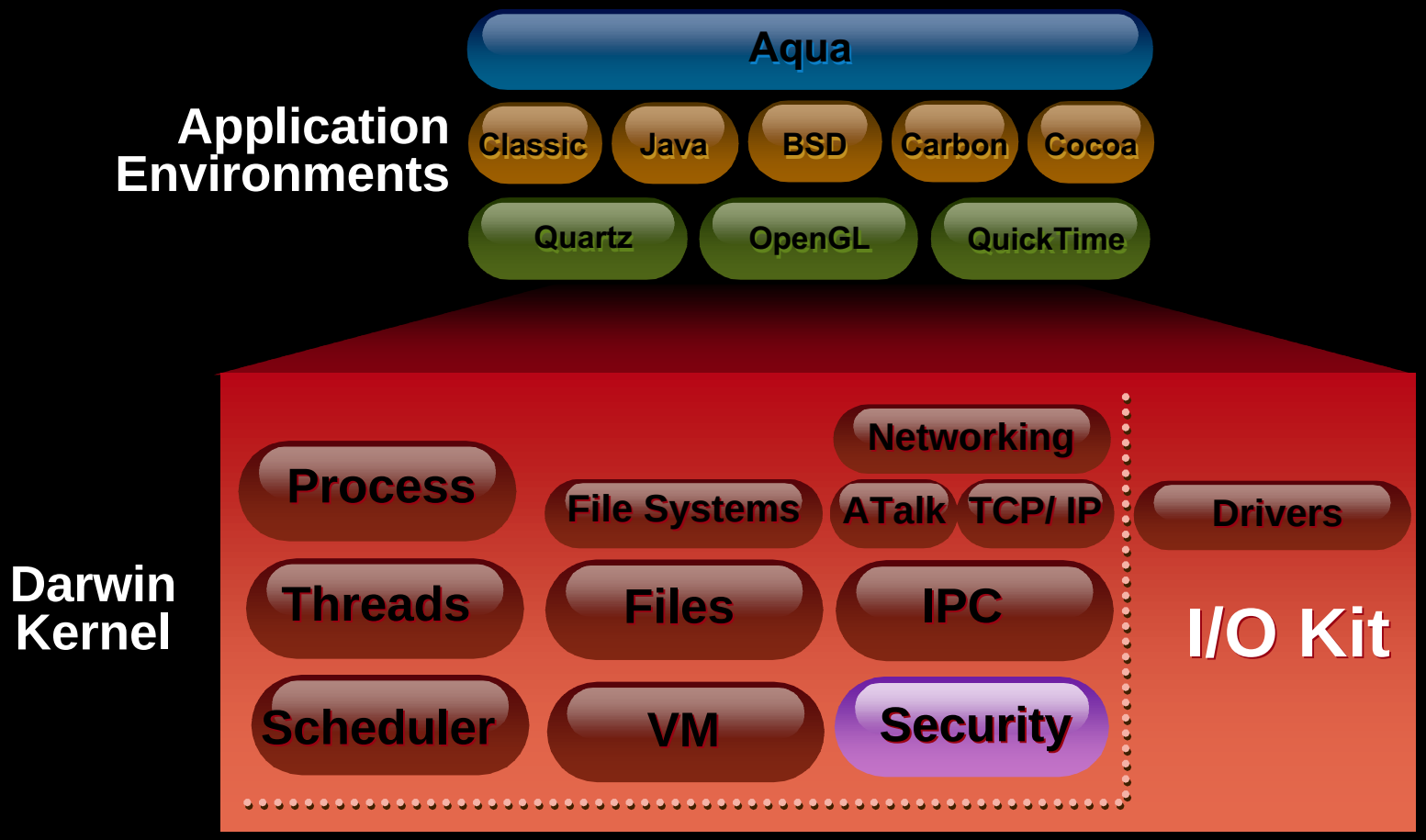
Tips

do not poll
use event model

Use MIG, not raw IPC

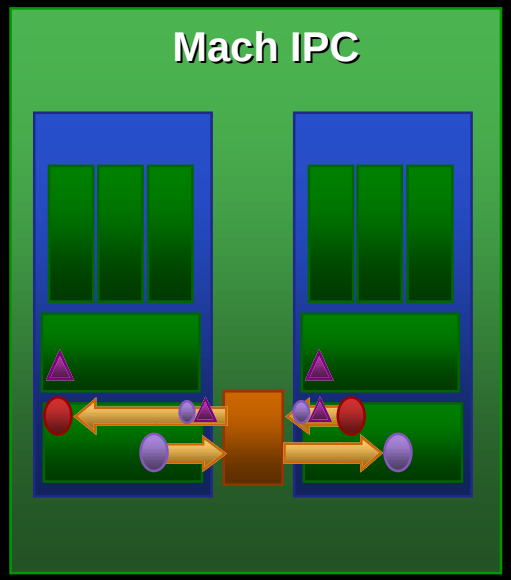
ports represent all manner of object references

Darwin Security Services



Mach Security Services

Implement mechanism—not policy



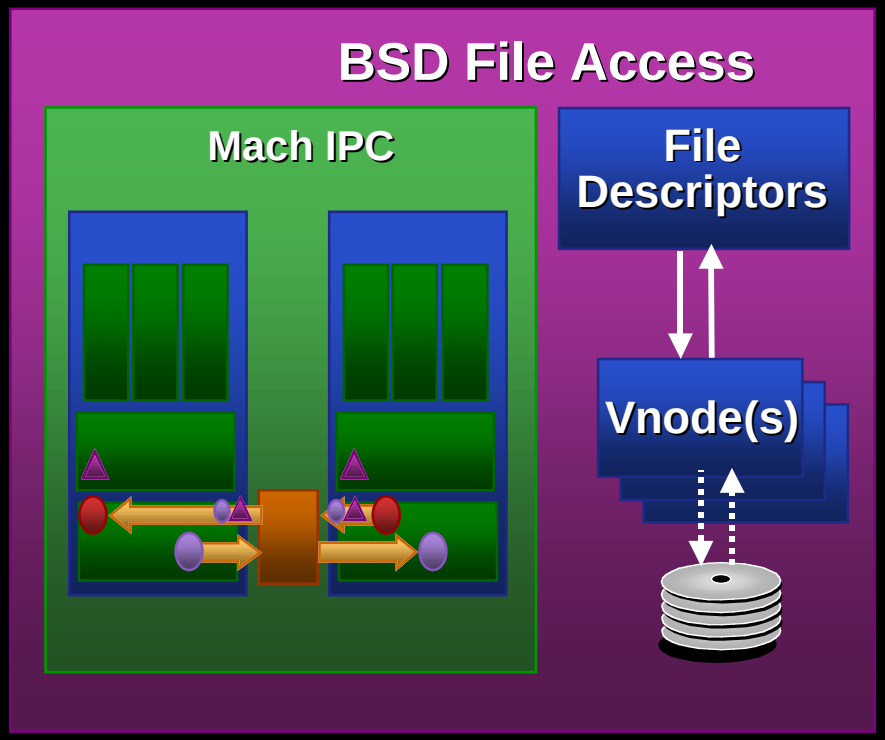
mach_port_t
restricted access
to port rights

control vs. normal access
on separate ports

security token tagging
on messages

BSD Security Services

Implement most policy decisions



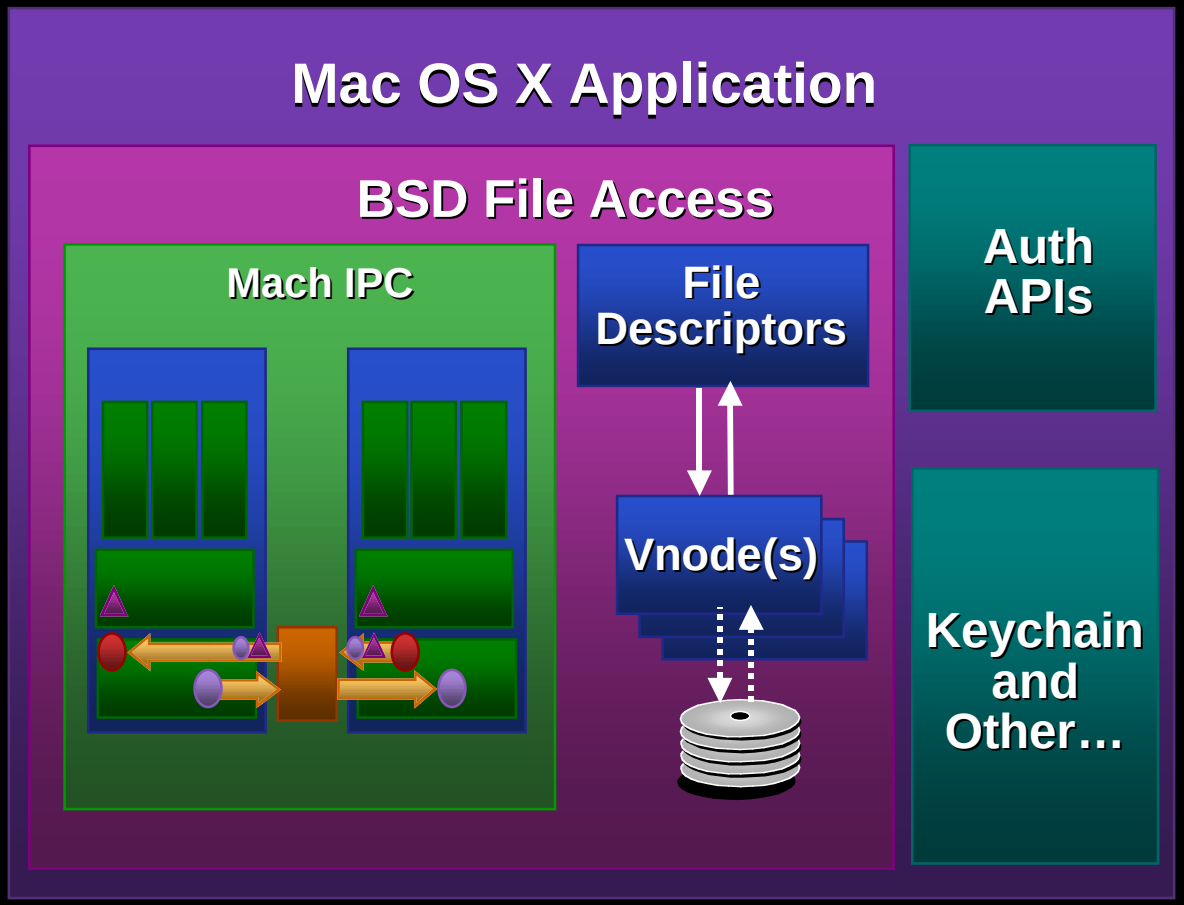
fd
maintains cached authorizations

file permissions
maintains data store
implements authorization

uid, euid, gid, egid
identification
real and effective

Application Security Services

Extend BSD/POSIX semantics

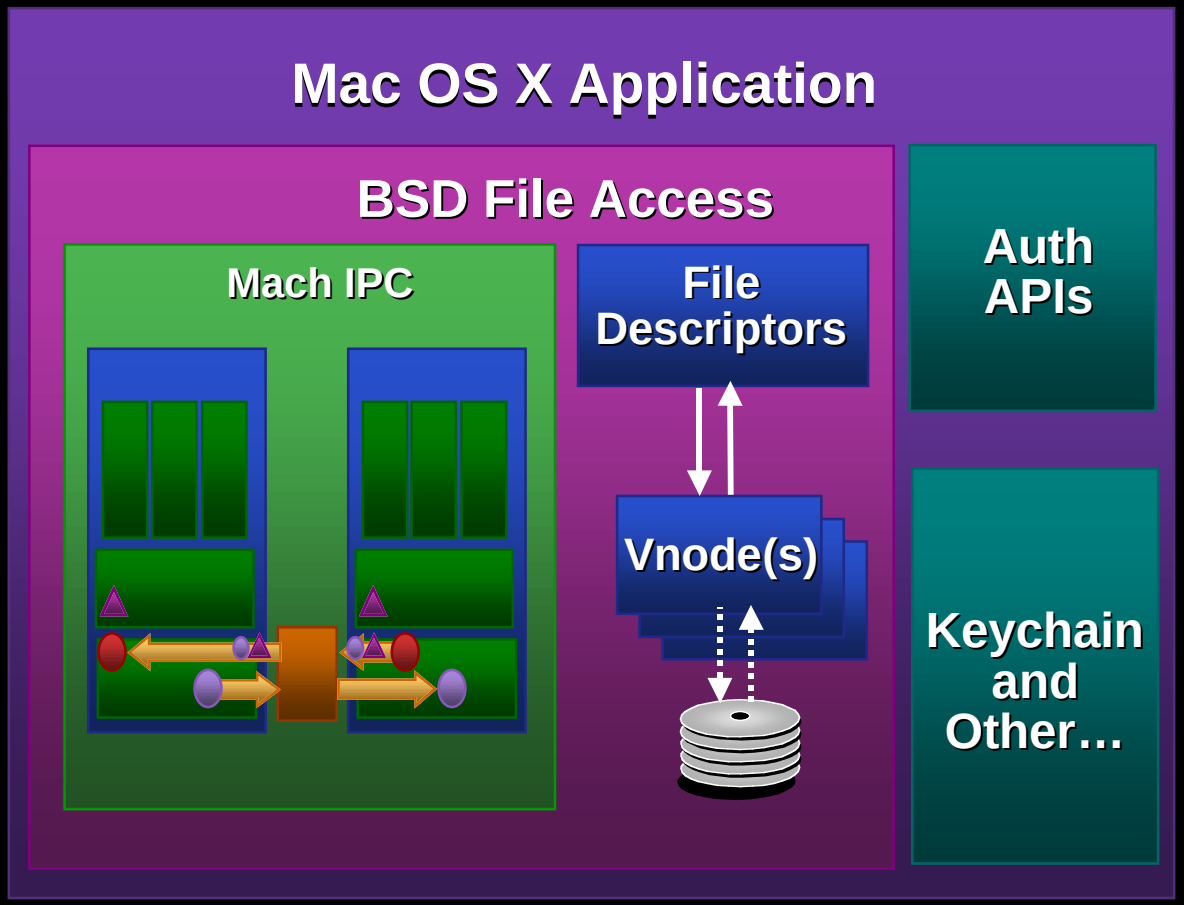


Authorization API

implements authorizations where BSD doesn't know how

Security Services

Tools and tips



Tools

fs_usage
vmmap
lsof

Tips

use capabilities (e.g., ports) in lieu of constant identification re-authorization

Where to Get More Info

Inside Mac OS X: Kernel Environment
On your Developer CD

Darwin Open Source
http://www.publicsource.apple.com

Mac OS X Homepage
http://developer.apple.com/macosx

The Design and Implementation of the BSD4.4 Operating System
by McKusick, et al.

http://developer.apple.com/events/urls.html

Roadmap

705 Apple Performance Tools	Room A2 **Thurs., 5:00pm**
706 Debugging on Mac OS X	Hall 2 **Fri., 9:00am**
139 Leveraging BSD Services in Mac OS X	Civic **Fri., 2:00pm**
FF121 Feedback Forum: Darwin	Civic **Fri., 5:00pm**

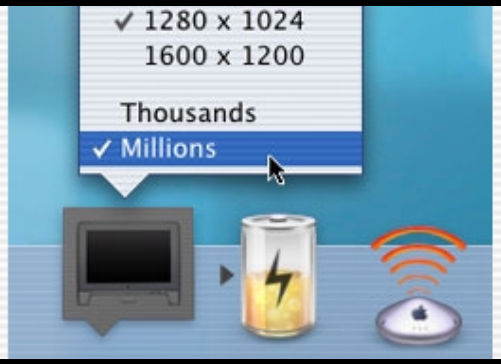
Who to Contact

Jason Yeo
Mac OS Technology Manager
Apple Worldwide Developer Relations
jason@apple.com

References

http://www.opensource.apple.com

Q&A



Jason Yeo
Mac OS Technology Manager
jason@apple.com

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001